UNIVERSIDADE DE LISBOA

FACULDADE DE CIÊNCIAS

DEPARTAMENTO DE MATEMÁTICA

Post-Quantum Cryptography: Lattice-Based Cryptography and Analysis of NTRU Public-Key Cryptosystem

Mestrado em Matemática

Rafael Torrado Monteiro

Dissertação orientada por:

Professor Doutor Paulo Alexandre Carreira Mateus
Professor Doutor Carlos Alberto Martins André

2016

Acknowledgements

First of all, I would like to thank the thesis advisors, Professor Paulo Mateus and Professor Carlos André, for always being available when I needed advise. Special gratitude goes to Professor Paulo Mateus, for suggesting the main topic for my thesis, post-quantum cryptography, which proved to be interesting and up-to-date; for spending his valuable time explaining the basic assumptions of classical and quantic computation that I had never learnt until then; for meeting me whenever I had doubts and, most of all, for accepting to advise my thesis without even knowing who I was and for always trusting and supporting me. My appreciation also goes to Professor Carlos André for his readiness to be my thesis advisor and to provide the information I needed throughout the study. Even though he was not too acquainted with the topic of this thesis, it was of uttermost importance to have an outside observer to examine the thesis. He also highly contributed with his opinion regarding the writing of this document.

Next, I want to thank my parents for their ongoing support and encouragement throughout my life and my university journey; also for providing me the opportunity to study abroad and move closer to the university, allowing me to have more time to study. A heartfelt thank you to my mother who helped me by reviewing the English writing of the thesis.

I also thank my grandparents for always being there, for always worrying about me and who are proud of my achievements.

Thank you to all my university colleagues in Mathematics, mainly Núria, Maria José, Alexandre, Bernardo, Raquel and Jorge, with whom I have spent some quality time throughout last years, be it studying, partying or travelling across Portugal. Thank you also to my other more advanced colleagues Tânia and Pedro, and a particular thank to Andreia and João for being able to make my day much more fun whem I'm with them.

I would like to take the opportunity to thank my two *course goddaughters* Ana and Catarina for sharing experiences and for exchanging ideas while simultaneously writing our theses.

Thank you Daniel and Raquel, for teaching me how to work with `C++` in a couple of hours. My gratitude to them and to Jéssica for the discussions on how to make this dissertation more appealing.

I would like to thank all those who have participated in the mathematics working group that João Fontinha and I have created. This working group had the objective of sharing knowledge between students and to disclose interesting subjects about Mathematics.

Thank you to my my former colleagues in Paris: Kévin, Geneviève and Steven for welcoming me within their community which is usually a closed circle. A special thanks to Maria, whom

I have shared my free time with, as well as my study time. I thank my roommates of *Maison du Portugal* in Paris, and especially Mrs. Ana Paixão, the director, for letting us make all the events that the Resident Committee (of which I was a member) had scheduled.

I would like to thank all my math teachers at Faculty of Sciences of University of Lisbon (FCUL), who have been always there for me when I had a doubt. I want to thank the professors from Université Pierre et Marie Curie (UPMC) in Paris, Professors Joseph Oesterlé and Leonardo Zapponi, who made me to enjoy much more the areas of pure algebra and cryptography.

Finally, I cannot fail to mention the huge help given by Mathematics, Cryptography and TeX-LaTeX StackExchange communities.

"Everything is possible.
The impossible just takes longer."
Dan Brown, Digital Fortress

Abstract

In 1994, Peter Shor has developed a quantum algorithm for integer factorization and the discrete logarithm problem, known as *Shor's algorithm*. This was a great finding in the quantum field, given that it makes essentially all currently used cryptographic constructions (such as RSA and ECC) easily breakable by a quantum computer. Since quantum computers are believed to phisically exist in the next few years, these encryption schemes will be no longer reliable, making the encrypted data compromised.

It is thus of crucial importance to build and analyze new quantum-resistant cryptographic schemes. These new cryptographic constructions should be efficient and secure enough to be used in practice and standardized for a large number of years, replacing the current ones if needed. In the last year, NIST (*National Institute of Standards and Technology*, from *U.S. Department of Commerce*) has announced its quest of finding such quantum-resistant cryptographic constructions.

The NTRU encryption scheme, developed by Jeffrey Hoffstein, Jill Pipher and Joseph Silverman (the same Silverman famous for his work on elliptic curves) in the 90's, presented in 1996 and published in 1998, is an example of a quantum-resistant proposal. Although it is not supported with a theoretical security proof, the scrutiny done since its presentation until now reveal that NTRU is secure and is a good successor for replacing the current constructions currently in use. It is a fact that there already exist some classical cryptosystems that are quantum-resistant. The McEliece cryptosystem (1978) is an example of such a quantum-resistant construction, but it is not good to be used in practice.

The theory behind the NTRU cryptosystem is very simple, therefore easy to understand, and NTRU can be very easily implemented and put in practice: both private and public keys are easy and quickly computable, having $O(N)$ length, making a clear difference between the length $O(N^2)$ of other cryptosystems considered as fast, and encrypting and decrypting with NTRU takes $O(N^2)$ operations, which makes it considerably faster than RSA and ECC.

To date, NTRU remains secure against both classical and quantum computers. The most effective attacks on NTRU are based on lattices, namely using lattice reduction algorithms, which goal is to find very short vectors in a lattice. One can apply the *LLL algorithm* (1982) of Arjen Lenstra, Hendrik Lenstra and László Lovász, which is recognized as a very important achievement in many areas of mathematics. It runs in polynomial time, but has an exponential approximation factor. This algorithm was improved over time. The most important improvement is due to Schnorr (1987), who introduced the *Blockwise-Korkine-Zolotarev algorithm*, also known as *BKZ algorithm*, which is to date the best lattice reduction algorithm to be put in practice. In 2011, Yuanmi Chen and Phong Q. Nguyen improved the BKZ algorithm, revising lattice security estimates, and created the *BKZ simulation algorithm*, permitting to predict both

the output and the running time of the BKZ algorithm without the need of running BKZ, since in high dimension with *blocksize* ≥ 45 the running time of BKZ is considerably long. There is a lot of investigation in this area, mainly because of its cryptographic applications.

This work can be divided in three parts: first, we present a slight and short introduction to quantum computing, namely the basics of quantum mechanics and the *Fourier Transform*; secondly, we give a sufficient groundwork on lattices, present the LLL and BKZ lattice reduction algorithms, the BKZ simulation algorithm as well, and give some examples of public-key encryption schemes; in third place, we give an analysis of the NTRU cryptosystem, focusing mainly on lattice-based attacks. Finally, as work to do in the future, we present an idea of a quantum algorithm for lattice reduction.

Keywords: Post-Quantum Cryptography; Quantum Computing; Lattice-based Cryptography; Lattice reduction algorithms; NTRU Public-Key Cryptosystem.

Resumo

Em 1994, Peter Shor desenvolveu um algoritmo quântico para a factorização de inteiros e para o problema do logaritmo discreto, fazendo de ambos RSA e ECC (*Elliptic Curve Cryptography*) facilmente quebráveis por computadores quânticos. Como se acredita que computadores quânticos irão fisicamente existir nos próximos poucos anos, quase toda a informação encriptada deixará de estar segura, e o impacto na vida prática das pessoas será no mínimo devastador. Aspetos como a comunicação entre pessoas, empresas e governos, operações bancárias, comércio eletrónico, passwords, entre muitos outros, estarão comprometidos. A criptografia de chave pública é hoje indispensável para indivíduos ou entidades comunicarem em segurança, logo, a existência de computadores quânticos constitui uma enorme ameaça para a nossa criptografia moderna.

Caso computadores quânticos venham realmente a existir, substituir os atuais criptossistemas em uso vai ser algo necessário, uma vez que os criptossistemas resistentes a computadores quânticos em uso estão muito longe de ser suficientemente eficientes para serem postos em prática. É portanto imperativo estudar criptossistemas candidatos a substituir os atuais em termos da sua eficiência, praticidade e segurança contra ambos computadores clássicos e quânticos. Em 2016, o NIST, *National Institute of Standards and Technology*, instituto do *Departamento do Comércio* dos EUA, anunciou a sua busca por tais construções criptográficas resistentes a computadores quânticos. Estas devem ser eficientes e suficientemente boas para ser adotadas por um largo número de anos. Este instituto prevê a chegada de computadores quânticos para 2030, o que revela que existe pouco tempo para estudar as eventuais propostas de substituição aos criptossistemas atuais. Além disso, esta substituição demorará certamente o seu tempo, de maneira que se deve começar já a investir na busca de construções criptográficas resistentes a computadores quânticos.

Desenvolvido por Jeffrey Hoffstein, Jill Pipher e Joseph Silverman (o mesmo Silverman famoso pelo seu trabalho em curvas elíticas) nos anos 90, apresentado na conferencia internacional de criptologia CRYPTO '96 e publicado em 1998, o *criptossistema NTRU* é um exemplo de um criptossistema resistente à computação quântica com potencial para vir a substituir os atuais criptossistemas clássicos em uso. Embora seja considerado um criptossistema baseado em reticulados, a sua descrição original não os usa, ainda que seja possível descrevê-lo inteiramente usando-os. Ainda assim, no artigo onde vem apresentado constam ataques baseados em reticulados que aproveitam certos aspetos que lhe são intrínsecos.

A teoria por detrás do criptossistema NTRU é bastante simples, dado que se baseia em álgebra polinomial e na redução módulo dois inteiros primos entre si. O criptossistema NTRU é, portanto, muito fácil de ser compreendido, implementado, e de ser posto em prática: ambas as chaves pública e privada são rapidamente computáveis, tendo comprimento $O(N)$, fazendo clara distinção com as chaves de comprimento $O(N^2)$ de outros criptossistemas ditos rápidos, tais como os criptossistemas McEliece e GGH, e encriptar e desencriptar com o criptossistema NTRU

leva $O(N^2)$ operações, o que faz do mesmo consideravelmente mais rápido que o criptossistema RSA e a criptografia baseada em curvas elíticas. Além disto, o criptossistema NTRU é considerado como sendo um criptossistema probabilístico, uma vez que usa um elemento aleatório para encriptar uma mensagem. Desse modo, uma mensagem pode ter múltiplas encriptações possíveis.

Desde a publicação do criptossistema NTRU, foram emitidos alguns relatórios técnicos, alguns consistindo em algoritmos que permitiram o criptossistema NTRU ganhar eficiência, outros descrevendo melhoramentos de ataques já existentes e também novos ataques e possíveis soluções para os contornar. A análise de segurança feita até hoje, incluindo por criptógrafos reconhecidos como Don Coppersmith, Johan Håstad, Andrew Odlyzho e Adi Shamir, deu ao criptossistema NTRU um estatuto de legitimidade e contribuiu para futura investigação, dado que o criptossistema NTRU se revelou interessante e promissor. Contudo, a análise realizada até hoje não implicou alterações profundas na estrutura do criptossistema NTRU, embora parâmetros mais largos tenham sido recomendados ao longo do tempo para alcançar os níveis de segurança desejados. Apesar de não ser acompanhado de uma prova de segurança, o criptossistema NTRU tem-se revelado seguro, uma vez que nenhum ataque quer clássico quer quântico com impacto significativo no criptossistema foi encontrado, o que revela que é uma boa alternativa aos criptossistemas clássicos em uso.

O nosso estudo do criptossistema NTRU começou por explorar a página da companhia *Security Innovation*, onde se pode encontrar uma breve introdução à criptografia pós-quântica e um vídeo que explica o porquê da mesma ser estudada. Esta página contém um vasto número de recursos, incluindo tutoriais, *surveys*, relatórios técnicos, resumos e artigos, assim como uma vasta lista de artigos que escrutinam o criptossistema NTRU. De modo a aprofundar o estudo do criptossistema NTRU, explorámos e investigámos uma grande parte da documentação existente nesta página. Existe, como se pode constatar, uma quantidade tremenda de documentação para analisar, e devido a limitações no tempo optámos por estudar o criptossistema NTRUEncrypt, começando pelas suas raízes.

Em 1997, Don Coppersmith e Adi Shamir fizeram a primeira análise de segurança ao criptossistema NTRU, em que descobriram que o método de ataque mais eficaz contra o criptossistema consistia em algoritmos de *redução de reticulados* com vista a encontrar vetores muito curtos numa classe particular de *reticulados*. Até à data, este é o método de ataque mais eficaz ao criptossistema NTRU. A segurança do criptossistema NTRU tem como base evidências empíricas, que mostra que encontrar vetores muito curtos num reticulado é um problema difícil, especialmente quando a dimensão do reticulado é muito grande.

O estudo dos reticulados, denominado *Geometria dos Números* por Minkowski, data anteriormente das suas aplicações criptográficas e é uma área que contribuiu para o desenvolvimento de muitas outras, tais como a física, análise, álgebra e geometria. Este é um tópico bastante estudado, principalmente devido às aplicações dos algoritmos de redução de reticulados. Como a segurança do NTRU depende principalmente de ataques baseados em reticulados, criámos um capítulo inteiro dedicado aos reticulados e a algoritmos de redução de reticulados. Um importante avanço nesta área foi o *algoritmo LLL* (por vezes chamado *algoritmo* L^3) de Lenstra, Lenstra e Lovász em 1982, desenvolvido antes dos reticulados serem considerados relevantes em criptografia. Este algoritmo permite encontrar um vetor moderadamente curto num reticulado em tempo polinomial. Com o passar do tempo, este algoritmo foi melhorado, dando origem ao *algoritmo BKZ* (*Blockwise-Korkine-Zolotarev*), que é até hoje na prática o melhor algoritmo

de redução de reticulados permitindo encontrar vetores muito curtos em reticulados. Este algoritmo deve-se a Schnorr (1987), no entanto não é executado em tempo polinomial. Portanto, encontrar vetores muito curtos num reticulado de larga dimensão continua a ser um problema em aberto. Em 2011, Yuanmi Chen e Phong Q. Nguyen criaram um algoritmo que permite simular o algoritmo BKZ, denominado *BKZ simulation algorithm*, podendo prever aproximadamente o *output* e o tempo de execução do algoritmo BKZ.

Este trabalho foi dividido em três partes. A primeira é uma pequena introdução à computação quântica, principalmente aos básicos da mecânica quântica e à *Transformada de Fourier*, uma vez que alguns leitores poderão não conhecer estes conteúdos. Além disso, algumas noções de computação quântica são mencionadas neste trabalho, pelo que faz todo o sentido incluir uma breve referência às mesmas. A segunda parte trata de reticulados em geral, apresenta os algoritmos de redução de reticulados LLL e BKZ (juntamente com o *BKZ simulation algorithm*) e também alguns exemplos de criptossistemas de chave pública que se baseiam em reticulados. A terceira parte revela uma análise do criptossistema NTRU, focando-se principalmente em ataques baseados em reticulados, que são os que de longe melhor se aplicam aos reticulados associados ao criptossistema NTRU. Como perspetiva de trabalho futuro, apresenta-se uma ideia de um algoritmo quântico para a redução de reticulados.

Palavras-Chave: Criptografia Pós-Quântica; Computação Quântica; Criptografia Baseada em Reticulados; Algoritmos de Redução de Reticulados; Criptossistema de Chave Pública NTRU.

Table of Contents

Chapter 1

Introduction

Developed by Jefferey Hoffstein, Jill Pipher and Joseph Silverman (the same Silverman famous for his work on elliptic curves) in the 90's, presented in the rump session of CRYPTO '96 and published in 1998, the *NTRU cryptosystem* is an example of a quantum-resistant cryptosystem with the potential to eventually replace the classical cryptosystems in use. This is thus an interesting cryptographic construction to be studied, but not only for this reason. Although it is considered as a *lattice-based cryptosystem*, its original description does not use lattices, even if it is possible to entirely describe it with lattices. Still, in its original publication [15] it is presented a lattice-based attack, taking advantage of key, message and random element spaces.

The theory behind the NTRU cryptosystem is very simple, since it is based in polynomial algebra and reduction modulo two relatively prime integers p and q. The NTRU cryptosystem is therefore easy to understand and, in addition, NTRU can be very easily implemented and put in practice: both private and public keys are easy and quickly computable, having $O(N)$ length, making a clear difference between the length $O(N^2)$ of other cryptosystems considered as fast, such as McEliece and GGH cryptosystems, and encrypting and decrypting with NTRU takes $O(N^2)$ operations, which makes it considerably faster than RSA and ECC (Elliptic Curve Cryptography) which encrypt and decrypt in $O(N^3)$ operations. Some of the advantages of NTRU (implementation, usability and performance levels) can be found in [33].

Besides, NTRU is considered as a probabilistic cryptosystem, since it uses a random element to encrypt a message, making the same message having multiple possible encryptions. Because of this, decryption may result in error, with probability depending on the choice of parameters. With elementary probability, one can study the probability of getting decryption errors with a chosen set of parameters. This is very important, since decryption errors can potentially reveal to an attacker some information about the message or the private key. However, to date this is not a problem, as we will remark in Chapters 3 and 4.

Since the publication of NTRU, *NTRU Cryptosystems, Inc.*, in 2009 acquired by *Security Innovation*, a software security company, has uploaded some technical reports, some of then consisting on algorithms which allowed NTRU to have better efficiency, and others describing improved or new attacks and possible solutions.

Despite of not having a security proof, the NTRU cryptosystem has revealed to be secure. The security analysis done until now, including by recognized cryptographers such as Don Coppersmith, Johan Håstad, Andrew Odlyzho and Adi Shamir, gave to NTRU a legitimate status and

contributed to further investigation, since NTRU has revealed to be interesting and promising. However, this analysis done until now did not suggested a profound change of the core idea and the scheme of NTRU, although larger parameters have been recommended in order to achieve the desired security levels. In 1997, Coppersmith and Shamir did the first analysis security of NTRU [6], in which they found that the most effective attacks on NTRU were based on lattices, using *lattice reduction algorithms* to find very short vectors in a special type of lattices, revising the security levels of the original parameter settings. So far, these presented lattice attacks and their amplifications remain the most effective on NTRU. Although lattices associated to NTRU have a special structure, the best attacks are also the best attacks on random lattices. However, it is not known if breaking NTRU is equivalent to being able to solve the underlying lattice problem for random lattices. The same goes for RSA: it is not known if classically breaking RSA is equivalent to being able to efficiently factorize.

The security of NTRU relies on empirical evidence, which shows that this underlying lattice problem of finding very short lattice vectors is hard, specially when the dimension of such lattices is very high. However, the fact of being based on experimental evidence is similar to the current cryptographic constructions in use. RSA and ECC are classically considered secure, since there is no classical efficient algorithm for integer factorization and the discrete logarithm problem. Furthermore, such efficient classical algorithm are not even in sight.

The study of lattices, called *Geometry of Numbers* by Minkowski, dates back before their application in cryptography, and is an area of mathematics that has contributed to many other areas, such as physics, analysis, algebra and geometry. This topic is intensively studied, specially because of lattice reduction applications. An important achievement in this area, and perhaps the most important, is the LLL algorithm (sometimes called L^3 algorithm) of Lenstra, Lenstra and Lovász, described in [21]. This algorithm is able to find a moderately short vector in a lattice in polynomial time, but has an exponential approximation factor, i.e., in high dimension, short vectors found by this algorithm are usually very far from the shorter vectors in the lattice. The LLL algorithm was improved over time, giving rise to the well-known BKZ algorithm, which is until today the best algorithm to be put in practice for finding very short vectors in a lattice in high dimension. However, the BKZ algorithm does not run in polynomial time, because it is based on mixing LLL and an exact algorithm. Hence as the dimension increases, its running time increases exponentially. Therefore, finding very short vectors in a lattice in sufficiently high dimension remains an open problem.

Since the physical existence of quantum computers is a threat to our modern cryptography, which is nowadays indispensable, being able to break almost all cryptosystems currently being used, such as RSA and ECC, it is of utmost importance to cryptanalyze the existing quantum-resistant proposals and to possibly build new more efficient and secure quantum-resistant cryptographic schemes. Once quantum computers come up, which is expected to be very soon, almost all encrypted data will be no longer safe, and the impact in practical life of all people would be devastating. Secure communications between people, governments and financial/non-financial companies, bank operations, passwords, e-commerce, data confidentiality and so on would be compromised. To prevent this catastrophic *armageddon* scenery, some agencies, such as NIST (*National institute of Standards and Technology*, from *U.S. Department of Commerce*) are already preparing the possible switch of current security protocols to quantum-resistant ones.

Therefore, post-quantum cryptography, or quantum-resistant cryptography, which is essentially the study of cryptographic schemes which are believed to be secure against both classical and

quantum algorithms, is a very important area of study.

1.1 Organization

Our study of the NTRU cryptosystem began with exploring the website [34]. On this page we can find a small introduction for post-quantum cryptography [38], and where we can find a short video which briefly explains why post-quantum cryptography is widely studied. Security Innovation's webpage contains a large number of resources [35], including tutorials, survey articles, technical reports, abstracts and articles, and contains as well a list of articles that scrutinize the NTRU cryptosystem [36]. In order to deepen the study of the NTRU cryptosystem we explored and investigated a great part of the documentation in these pages. There is of course a tremendous amount of documentation to analyse, and due to time limitations we choose to study NTRUEncrypt, beginning by its roots.

As the security analysis of NTRU depends primarily on lattice-based attacks, we made an entire chapter focusing on lattices and and lattice reduction algorithms. In this same chapter, we give other examples of lattice-based cryptosystems, such as the GGH/HNF and the LWE-Based cryptosystems. In the following chapter that we reserved for the security analysis of NTRUEncrypt, we give special attention to lattice attacks, the ones that better and by far applies to NTRU lattices. We also study other not so effective attacks, but still important and interesting. Furthermore, we compare NTRU with other well known cryptosystems, such as RSA, ECC and McEliece. We emphasize that the study of NTRUEncrypt that we make in this work is based on the original construction given in [13] and [15], and that the current construction is quite different, adjusted to resist, for example, to chosen ciphertext attacks, so many of the presented attacks are no longer functional. Nevertheless, the underlying lattice problem is the same.

We give a small introduction to quantum computing, that can be found in [2], [26] and [48]. Although this introduction is not very detail-oriented, we have decided to include a chapter devoted to this subject since some readers may not be familiar with these contents. Besides, some notions of quantum computing are mentioned in this work, and it made perfect sense to include them here. In the last chapter, we present an idea of a quantum algorithm for lattice reduction and also some prospects for future work.

1.2 Contributions

The major contributions of this work, although very modest, are listed in the following topics:

1. In Section 3, we proved all that we considered necessary. In particular, in Corollary 1 of Section 3.2.1 we proved the third statement, which was already known, still we did not find a proof for it;

2. In Section 3.2.2, we try to explain in detail the BKZ algorithm more clearly than in [5] and [30]. We did the same for the BKZ simulation algorithm, which is not entirely explained in [5]. Although [30] has served as a reference, there are some unclear statements that we try to explain in a better way, such as the volume calculation of local projected lattices;

3. In Section 3.3.3, we try to explain in more detail some omitted calculations in [2];

4. In the security analysis of NTRU, we explain the need for taking parameters p and q relatively prime, since all references that describe this only mention the case $p \mid q$;

5. In Section 4.1.3, we state that $|\ |_2$ and $|\ |_\infty$ are both seminorms, which is not stated in all references that we found about this;

6. We implemented in Matlab Proposition 6 of Section 4.1.3, giving different constants γ_1 and γ_2 than the ones given in [15];

7. In Section 4.1.6, we give an example for the meet-in-the-middle attack;

8. In Section 4.1.9, we explain in a better way how constants c_h and c_m are found, given that in [13] and [15] there were missing some steps;

9. Finally, in Chapter 5 we present an idea for a quantum algorithm to solve SVP in generic lattices, although it has to be studied in more detail and with more time.

Chapter 2

Quantum Computing

The security of classical public-key schemes relies on the hardness of solving certain number theory problems. However, with the introduction of quantum algorithms and quantum computers, some of them are no longer secure. However, there are some known cryptographic constructions that resist to such quantum algorithms, so even if quantum computers become a reality, information security is preserved.

Such quantum-resistant constructions, such as *hash-based cryptography*, *code-based cryptography*, *lattice-based cryptography* (which will be our focus), *multivariate-quadratic-equations cryptography* and *secret-key cryptography*, are believed to be secure against both classical and quantum computers, based on the hardness of solving the underlying number theory problems in both classical and quantum settings.

For all these classes of quantum-resistant constructions, it is unknown how to apply the existing quantum algorithms (such as *Shor's algorithm*[1], for integer factorization and the discrete logarithm problem, and *Grover's algorithm*[2], for searching a certain item in an unordered space with a finite number of elements) and making new algorithms for breaking any of these classes of constructions seems to be difficult.

Since quantum computers are expected to exist in few years from now, it is of extreme importance to study those cryptographic constructions classes. Ideally, the goal would be to find a devastating attack for a construction, such as Shor's algorithm for factorization, making it totally insecure and therefore useless, or more modestly to find an attack that forces to large key sizes, such as Grover's algorithm.

The security of some cryptosystems relies exclusively on the experimental evidence done, but others have theoretical security proofs which makes them considered as being more secure than the previous ones. However, these security proofs are based on the hardness of solving some number theory problem, where it is unknown if this problem is actually hard to solve. Therefore, the security of these cryptosystems is due to the lack of knowledge of effective attacks on their underlying problem.

[1]The best classical algorithm for integer factorization runs in $O(\exp(c(\log N)^{1/3}(\log\log N)^{2/3}))$, where N is the integer to factorize and c is a constant. It is known as *number field sieve*. Shor's algorithm breaks factorization in $O((\log N)^2(\log\log N)(\log\log\log N))$ time.

[2]Grover's algorithm has time complexity $O(\sqrt{N})$, where N is the number of elements in the search space.

If quantum computers become a reality, replacing the current cryptographic constructions will be something necessary, since the current quantum-resistant schemes are too far of being efficient to be put in practice. For example, the McEliece cryptosystem could never be used in practice, since its keys would be of a few million bits. It is thus important to improve the known quantum-resistant cryptographic constructions in terms of their efficiency, security and practicality, or if necessary to build new secure, efficient and practical quantum-resistant schemes.

Still, no one knows if this replacement will be actually necessary. No one actually knows if the study of post-quantum cryptography is needed, since no one knows if quantum computers will physically exist or not. However, it is believed that it is a question of time and money until a large quantum computer appears (NIST believes that they will be built by 2030 for the price of one billion dollars). Such large quantum computers would be able to break RSA-2048 in a few hours, so larger key sizes for all current cryptographic constructions would be required, and by this way the efficiency of all current cryptographic constructions would collapse.

Briefly, quantum computers are a threat to our modern cryptography, and we are not ready to replace the currently used cryptosystems by quantum-resistant and efficient ones, since more study is required to build trust in these constructions and since it would take a considerable time to exchange them. Although no one can assure that quantum computers will come up by the next few years, it is of critical importance to develop research in post-quantum cryptography, to not be caught off guard in the case of quantum computers suddenly arise.

2.1 Classical Cryptography and Quantum Computing

In 1994, Peter Shor developed a quantum algorithm for integer factorization and the discrete logarithm problem, making RSA and Diffie-Hellman key exchange breakable by a quantum computer. Because of this, new problems difficult to solve by quantum computers are required in order to build new cryptographic schemes. However, this difficulty assumption is based on experimental evidence, which means that whatever cryptographic construction is chosen, its security depends on the security of the underlying problem, which is entirely based on empirical evidence. For example, RSA is considered secure in the classical setting because factoring is computationally hard. As we will observe, NTRU is considered secure in the classical setting because its underlying lattice problem is computationally hard, and is considered secure in the quantum setting because there is no efficient quantum attacks against it.

The design of new quantum-resistant cryptographic schemes can be a difficult assignment, specially because of quantum computation rules. It is believed that NP-complete problems do not induce good cryptographic constructions, and on the other hand it is not believed that quantum computers can efficiently solve NP-complete problems. In addition, applying the known results of quantum computing in cryptanalysis is, in practice, very difficult. For example, Shor's algorithm uses *Quantum Fourier Transforms* (QFTs) to break the integer factorization problem in the quantum setting, but QFTs seem not to be applicable to other problems, such as lattice-based ones. This is why Shor's algorithm is a remarkable result in cryptography. Today, there are multiple efficient quantum algorithms for solving some problems in the most diverse areas, but the list of such problems is still small when compared with the list of problems which lack of an efficient quantum algorithm.

The following table, inspired on a table of [2], summarizes the current state of public-key cryptography:

Cryptosystem	**Broken by Quantum Algorithms?**
RSA public-key encryption	Yes
Diffie-Hellman key-exchange	Yes
Elliptic curve cryptography	Yes
McEliece public-key encryption	Not yet
NTRU public-key encryption	Not yet
Lattice-based public-key encryption	Not yet

As said before, if the need to change current cryptosystems for quantum-resistant ones arises, the candidates must be extremely reliable and efficient to be standardized for a large number of years. However, the currently known cryptosystems that are quantum-resistant have larger key sizes when compared to classical ones, as we will see in Chapter 4. Although many of these cryptosystems date are before 2000, so more scrutiny is needed. NIST is now preparing for a quantum-resistant transition, opening on Fall 2016 a call for proposals of quantum-resistant public-key, digital signature and key exchange protocols. The deadline for submissions is November 2017, and in the next three to five years there will be an analysis phase, where the proposals will be subject of public scrutiny, in which NIST will report its findings. After that time, NIST may decide whether to standardize or not the considered algorithms. This timeline may potentially be changed if the developments in the field justify so.

It is worth mentioning that this is not a competition. After the public analysis phase, NIST may choose more than one presented construction to be standardized, this in the ideal case, since there is a possibility of none being. The received proposals will be evaluated based on three elements: security, cost (computational efficiency and memory requirements) and the algorithm itself and its implementation characteristics (ease of implementation, ease of use, simplicity, among others). Therefore, there is a possibility of changing the current security protocols in the next few years, so agencies should be prepared for this transition.

2.2 Quantum Mechanics

These next sections intend to introduce the basics of quantum mechanics, memory and computing, showing what is and what is not allowed in quantum computing.

We define a *pure quantum state* $|\phi\rangle$ (or simply a *state*) as a *superposition* of N classical states. Such classical states will be denoted by $|1\rangle, \ldots, |N\rangle$, and we write

$$|\phi\rangle = \alpha_1|1\rangle + \cdots + \alpha_N|N\rangle,$$

where $\alpha_1, \ldots, \alpha_N \in \mathbb{C}$. The value $\alpha_i \in \mathbb{C}$ is called the amplitude of $|i\rangle$ in $|\phi\rangle$.

For example, consider the classical states $|A\rangle$ and $|B\rangle$ defined as *dead* and *alive*, respectively. Hence a cat in the quantum state $\alpha|A\rangle + \beta|B\rangle$ is dead and alive at the same time (dead with

amplitude α and alive with amplitude β).

Now that we have defined what quantum states are, we will describe what kind of operations we can apply to them. There are only two possibilities: either measure the quantum state, or make it evolve to another quantum state by applying a unitary operator.

2.2.1 Measurement

Consider the cat's example given above. If the cat is in the quantum state $|\Psi\rangle = \alpha|A\rangle + \beta|B\rangle$, we cannot actually *see* if the cat is dead or not, since it is dead and alive at the same time.

Let us define what is measuring a quantum state $|\phi\rangle = \alpha_1|1\rangle + \cdots + \alpha_N|N\rangle$. We can define it as a correspondence that when given a quantum state $|\phi\rangle = \alpha_1|1\rangle + \cdots + \alpha_N|N\rangle$, outputs a classical state $|j\rangle$ with probability $|\alpha_j|^2$. Hence,

$$\sum_{j=1}^{N} |\alpha_j|^2 = 1,$$

which implies that $\|(\alpha_1, \ldots, \alpha_N)\| = 1$.

Suppose that the cat is dead with probability $1/3$ and is alive with probability $2/3$. Then,

$$|\Psi\rangle = \frac{1}{\sqrt{3}}|A\rangle + \frac{\sqrt{2}}{\sqrt{3}}|B\rangle.$$

A priori, the result of measuring $|\Psi\rangle$ is unknown, but the cat is more likely to be alive. When we measure $|\Psi\rangle$, we will see the classical state $|A\rangle$ or $|B\rangle$, and know if the cat is *really* dead or alive.

When we measure a quantum state $|\phi\rangle$ and see the resultant classical state $|j\rangle$, we say that $|\phi\rangle$ has *collapsed* and no more operations can be done to the result of the measurement.

2.2.2 Unitary Evolution

A quantum state $|\phi\rangle = \alpha_1|1\rangle + \cdots + \alpha_N|N\rangle$ can change to another quantum state $|\psi\rangle = \beta_1|1\rangle + \cdots + \beta_N|N\rangle$ in time, but only linear operations are allowed on quantum states, i.e., viewing $|\phi\rangle$ and $|\psi\rangle$ as N-dimensional vectors $(\alpha_1, \ldots, \alpha_N)$ and $(\beta_1, \ldots, \beta_N)$ respectively, the change of $|\phi\rangle$ into $|\psi\rangle$ is given by the matrix multiplication

$$U \begin{pmatrix} \alpha_1 \\ \vdots \\ \alpha_N \end{pmatrix} = \begin{pmatrix} \beta_1 \\ \vdots \\ \beta_N \end{pmatrix},$$

where $U \in \mathbb{C}^{N \times N}$. By linearity, we have

$$|\psi\rangle = U|\phi\rangle = U\left(\sum_{i=1}^{N} \alpha_i|i\rangle\right) = \sum_{i=1}^{N} \alpha_i U|i\rangle.$$

By measuring $|\psi\rangle$, we must obtain

$$\sum_{j=1}^{N} |\beta_j|^2 = 1,$$

so U preserves the euclidean norm of vectors, and therefore must be unitary. Since a unitary operator has an inverse, we conclude that all non-measuring operation done to a quantum state is *reversible*, i.e., we can always *undo* a non-measuring operation by applying the inverse of the operator:

$$\begin{pmatrix} \alpha_1 \\ \vdots \\ \alpha_N \end{pmatrix} = U^{-1} \begin{pmatrix} \beta_1 \\ \vdots \\ \beta_N \end{pmatrix}.$$

It is clear that measuring a quantum state is not reversible, because we cannot reconstruct the vector of amplitudes $(\alpha_1, \ldots, \alpha_N)$.

2.3 Quantum Memory

In contrast with classical computers which use the *bit* as basic unit of information, quantum computers work with *quantum bits* (or simply *qubits*). A qubit is a superposition of the classical bits 0 and 1, say $\alpha|0\rangle + \beta|1\rangle$, where $|\alpha|^2 + |\beta|^2 = 1$. Hence every qubit can be viewed as a unit vector in the Hilbert space

$$\mathcal{H} = \mathcal{H}_1 = \mathbb{C} \oplus \mathbb{C}.$$

We will consider its standard basis, consisting on the vectors $(1, 0)$ and $(0, 1)$, and assign $|0\rangle$ to $(1, 0)$ and $|1\rangle$ to $(0, 1)$.

We define a *n-qubit* as a superposition of all possible n-bits and write it as

$$\sum_{i_1,\ldots,i_n \in \{0,1\}} \alpha_{i_1,\ldots,i_n} |i_1 \cdots i_n\rangle,$$

where

$$\sum_{i_1,\ldots,i_n \in \{0,1\}} |\alpha_{i_1,\ldots,i_n}|^2 = 1.$$

Similarly, every n-qubit can be viewed as a unit vector in the Hilbert space

$$\mathcal{H}_n = \mathcal{H}^{\otimes n},$$

which has as standard basis the vectors $|i_1\rangle \otimes \cdots \otimes |i_n\rangle$, where $i_1, \ldots, i_n \in \{0, 1\}$. Hence we assign $|i_1 \cdots i_n\rangle$ to $|i_1\rangle \otimes \cdots \otimes |i_n\rangle$.

To ease notation, sometimes we write $|i_1 \cdots i_n\rangle$ as $|(i_1 \cdots i_n)_{10}\rangle$, where $(\ )_{10}$ is the representation

in base 10. Therefore a n-qubit can be written as

$$\alpha_0|0\rangle + \alpha_1|1\rangle \cdots + \alpha_{2^n-1}|2^n - 1\rangle,$$

where

$$\sum_{j=0}^{2^n-1} |\alpha_j|^2 = 1.$$

2.4 Elementary Gates

A *gate* is a unitary operator that acts on n-qubits with n small. It is the quantum counterpart of the classical logic gates AND, OR and NOT. We will represent these gates with matrices, so recall the definition of a kronecker product of matrices.

The simplest examples are the 1-qubit gates

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \quad \text{and} \quad Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix},$$

called *bitflip gate* and *phaseflip gate*, respectively. We have

$$\begin{array}{ll} X|0\rangle = |1\rangle & \\ X|1\rangle = |0\rangle & \end{array} \quad \text{and} \quad \begin{array}{l} Z|0\rangle = |0\rangle \\ Z|1\rangle = -|1\rangle. \end{array}$$

Another 1-qubit gate is the *phase-gate*

$$R_\phi = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\phi} \end{pmatrix},$$

where

$$\begin{aligned} R_\phi|0\rangle &= |0\rangle \\ R_\phi|1\rangle &= e^{i\phi}|1\rangle. \end{aligned}$$

The *Hadamard gate* is one of the most important gates, since when we measure a quantum state, all classical states have equal probability of being observed. In Chapter 5 we will use these such a gate for our lattice reduction quantum algorithm attempt. The 1-qubit Hadamard gate is defined as

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix},$$

that is,

$$\begin{aligned} H|0\rangle &= \tfrac{1}{\sqrt{2}}|0\rangle + \tfrac{1}{\sqrt{2}}|1\rangle \\ H|1\rangle &= \tfrac{1}{\sqrt{2}}|0\rangle - \tfrac{1}{\sqrt{2}}|1\rangle. \end{aligned}$$

We remark that

$$H^2|0\rangle = |0\rangle$$
$$H^2|1\rangle = |1\rangle.$$

Similarly, we can define the n-qubit Hadamard gate recursively as

$$H_n = H_1 \otimes H_{n-1} = \frac{1}{\sqrt{2}} \begin{pmatrix} H_{n-1} & H_{n-1} \\ H_{n-1} & -H_{n-1} \end{pmatrix}.$$

Finally, to complete this list of examples, we have the *controlled-not gate* CNOT, which is a 2-qubit gate:

$$\text{CNOT} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}.$$

We have

$$\text{CNOT}|0\rangle|0\rangle = |0\rangle|0\rangle$$
$$\text{CNOT}|0\rangle|1\rangle = |0\rangle|1\rangle$$
$$\text{CNOT}|1\rangle|0\rangle = |1\rangle|1\rangle$$
$$\text{CNOT}|1\rangle|1\rangle = |1\rangle|0\rangle,$$

hence

$$\text{CNOT}|0\rangle|b\rangle = |0\rangle|b\rangle$$
$$\text{CNOT}|1\rangle|b\rangle = |0\rangle|1-b\rangle.$$

2.5 The Fourier Transform

2.5.1 The Classical Discrete Fourier Transform

For a fixed N, the *Discrete Fourier Transform* (DFT) is defined as the unitary operator whose matrix is

$$F_N = \frac{1}{\sqrt{N}} \left(\omega_N^{jk}\right)_{j,k \in \{0,\ldots,N-1\}}$$

where $\omega_N = e^{2\pi i/N}$. If $N = 2$, we get the 1-qubit Hadamard gate

$$F_2 = H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}.$$

For any vector $v \in \mathbb{R}^N$, it is said that $\hat{v} = F_N v$ is the *Discrete Fourier Transform of* v. Doing the usual matrix-vector multiplication, we get

$$\hat{v}_j = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} \omega_N^{jk} v_k.$$

2.5.2 The Fast Fourier Transform

The previous naive calculation of $\hat{v} = F_N v$ takes $O(N)$ operations for each coordinate, thus $O(N^2)$ are required to compute $\hat{v}$. However, there is a method to efficiently compute $\hat{v}$ that takes only $O(N \log N)$ operations, called the *Fast Fourier Transform* (FFT). When N is too large, this speedup is shown to be of great importance in practice.

From now on, we will suppose that N is a power of 2. This is not really necessary, but the following computations get easier. The core idea of FFTs is to build a recursive computation method:

$$\begin{aligned}
\hat{v} &= \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} \omega_N^{jk} v_k \\
&= \frac{1}{\sqrt{N}} \left(\sum_{k \text{ even}} \omega_N^{jk} v_k + \sum_{k \text{ odd}} \omega_N^{jk} v_k \right) \\
&= \frac{1}{\sqrt{N}} \left(\sum_{k \text{ even}} \omega_N^{jk} v_k + \omega_N^{j} \sum_{k \text{ odd}} \omega_N^{j(k-1)} v_k \right) \\
&= \frac{1}{\sqrt{N}} \left(\sum_{k \text{ even}} \omega_{N/2}^{jk/2} v_k + \omega_N^{j} \sum_{k \text{ odd}} \omega_{N/2}^{j(k-1)/2} v_k \right) \\
&= \frac{1}{\sqrt{2}} \left(\frac{1}{\sqrt{N/2}} \sum_{k \text{ even}} \omega_{N/2}^{jk/2} v_k + \omega_N^{j} \frac{1}{\sqrt{N/2}} \sum_{k \text{ odd}} \omega_{N/2}^{j(k-1)/2} v_k \right) \\
&= \frac{1}{\sqrt{2}} (\widehat{v^{\text{even}}}_j + \omega_N^j \widehat{v^{\text{odd}}}_j).
\end{aligned}$$

where $\widehat{v^{\text{even}}}$ is the FFT of the vector of even-numbered entries of v and $\widehat{v^{\text{odd}}}$ is the FFT of vector of odd-numbered entries of v.

The required time to compute the FFT of $\hat{v}$ is therefore $T(N) = 2T(N/2) + O(N)$, and using Wolfram|Alpha we can conclude that $T(N) = O(N \log N)$. For the same reason, the *inverse Fourier Transform*, whose matrix is $F_N^{-1} = F_N^*$, is also computed in $O(N \log N)$ operations.

Let us give an example of application of FFTs. Let $P(X), Q(X) \in \mathbb{R}[X]$ such that $\deg(P), \deg(Q) \leq d$. We will give an algorithm for multiplying these two polynomials in $O(d \log d)$ steps:

1. Supposing that

$$P(X) = \sum_{i=0}^{d} a_i X^i \quad \text{and} \quad Q(X) = \sum_{j=0}^{d} b_j X^j,$$

write the product $P(X)Q(X)$ as

$$P(X)Q(X) = \sum_{l=0}^{2d} \left(\underbrace{\sum_{k=0}^{2d} a_k b_{l-k}}_{c_l} \right) X^l,$$

where $a_k = b_k = 0$ for $k > d$ and $b_{l-k} = 0$ if $k > l$;

2. Define $N = 2d + 1$ and set $a = (a_0, \ldots, a_d, 0, \ldots, 0)$ and $b = (b_0, \ldots, b_d, 0, \ldots, 0) \in \mathbb{R}^N$;

3. Compute $\hat{a}$ and $\hat{b}$;

4. Compute $\widehat{a * b}$, where $*$ is a convolution product defined by

$$(a * b)_l = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} a_j b_{l-j \pmod N}.$$

There are two things to remark about this convolution product:

(a) $(\widehat{a * b})_l = \hat{a}_l \cdot \hat{b}_l$;

(b) $c_l = \sqrt{N}(a * b)_l$;

5. Apply the inverse FFT to $\widehat{a * b}$ to get $a * b$;

6. Multiply $a * b$ by $\sqrt{N}$ in order to get the coefficients c_l.

2.5.3 The Quantum Fourier Transform

When $N = 2^n$, the matrix F_N can be can be seen as a quantum operation on quantum states as in Section 2.2.2, given that F_N is unitary. The n-qubit unitary operator with matrix F_N,

$$|k\rangle \mapsto F_{2^n}|k\rangle = \frac{1}{\sqrt{2^n}} \sum_{j=0}^{2^n-1} \omega_{2^n-1}^{jk} |j\rangle,$$

is called the *Quantum Fourier Transform* (QFT).

2.6 The No-Go Theorems

In the quantum setting, some situations of classical computation are not possible. Some of these situations, often called *no-go theorems*, can be stated as in the following theorems. We only present two well-known no-go theorems, since we are not going to need them further on. We also remark that we only prove the first presented theorem, since the other proof is very similar.

Theorem 1 (No-cloning)**.** *There is no unitary matrix U such that, for all quantum state $|\phi\rangle$,*

$$U|\phi\rangle|0\rangle = |\phi\rangle|\phi\rangle.$$

Proof. Let $|\phi\rangle = \alpha|0\rangle + \beta|1\rangle$ be an arbitrary quantum state. Suppose that there exists a unitary matrix U such that

$$U|\phi\rangle|0\rangle = |\phi\rangle|\phi\rangle.$$

Therefore,

$$\begin{aligned} U|\phi\rangle|0\rangle &= U\left(\alpha|0\rangle + \beta|1\rangle\right)|0\rangle \\ &= \alpha U|0\rangle|0\rangle + \beta U|1\rangle|0\rangle \\ &= \alpha|0\rangle|0\rangle + \beta|1\rangle|1\rangle \\ &= \alpha|00\rangle + \beta|11\rangle \end{aligned}$$

and

$$\begin{aligned} U|\phi\rangle|0\rangle &= |\phi\rangle|\phi\rangle \\ &= (\alpha|0\rangle + \beta|1\rangle)(\alpha|0\rangle + \beta|1\rangle) \\ &= \alpha^2|00\rangle + \alpha\beta(|01\rangle + |10\rangle) + \beta^2|11\rangle. \end{aligned}$$

Hence $\alpha\beta = 0$, so $\alpha = 0$ or $\beta = 0$, which is a contradiction. □

Theorem 2 (No-deletion)**.** *There is no unitary matrix U such that, for all quantum states $|\phi\rangle$ and $|\psi\rangle$,*

$$U|\phi\rangle|\phi\rangle|\psi\rangle = |\phi\rangle|0\rangle|\psi'\rangle.$$

Chapter 3

Lattice-based Cryptography

Lattice-based cryptography is a relatively new area of research that started with the work of Ajtai, who was the first using lattices for cryptography purposes. Despite of being recent, this area is nowadays very studied.

Although it has not been proved, it is believed that lattice-based cryptographic constructions are quantum-resistant. In addition to that, these constructions demarcate up for their simplicity, efficiency in implementation, and for their security proofs based on worst-case hardness. This means that breaking such a cryptosystem (even with small probability) would imply solving any instance of the underlying lattice problem.

The presumed hardness of certain lattice problem is the base of lattice-based cryptosystems. One of them is the *shortest vector problem* (SVP), whose statement is: given an arbitrary basis for a lattice, find the shortest nonzero vector in the lattice. Notice that referring to *the* shortest vector in a lattice is an abuse of terminology, since this vector is required not to be the trivial one (the zero vector) and since there are at least two *shortest* vectors in a lattice (if v is the *shortest* vector in a lattice, then $-v$ is also a *shortest* vector in the same lattice).

For solving such lattice problems, there is a well known and widely studied algorithm developed in 1982 by Lenstra, Lenstra and Lovász, called *LLL algorithm*, with many applications, such as factoring polynomials over $\mathbb{Q}$ [21] and break special cases of RSA. For that reason, this algorithm (and others) has a major role in public-key cryptanalysis. The LLL algorithm runs in polynomial time for SVP and achieves an approximation factor of $2^{O(n)}$ where n is the dimension of the lattice.

Later in 1987, Schnorr presented the *BKZ algorithm*, which is an extension of the LLL algorithm for solving SVP that leads to shorter vectors. However, the running time of this algorithm increases significantly with the choice of better parameters. Still, this is the best lattice reduction algorithm known in practice for high dimension.

There is also a variant of Schnorr's algorithm, the *slide reduction algorithm* of Gama and Nguyen [9], which is in theory the best approximation algorithm, yet outperformed by BKZ.

The LLL and BKZ algorithms are approximation algorithms, but we can find some exact algorithms that output the shortest vector of the lattice (or at least nearly shortest vectors). As we will see, BKZ algorithm has a subroutine algorithm called *Enum*, which is an enumeration

algorithm that is exact. Having these two types of algorithms, approximate and exact ones, we can combine both as evidenced by the BKZ algorithm.

Exact algorithms for lattice problems generally run in at least exponential time. The best known today is given in [1], exponential in time and space. Therefore, exact algorithms are impractical when the lattice dimensions are sufficiently high. Other algorithms run in polynomial space, yet run in $2^{O(n \log n)}$ time. With no polynomial time algorithm known until today, it is believed that lattice-based cryptographic constructions are secure.

Such cryptographic constructions can be divided into practical ones and ones with strong security proofs. The reason why this division is possible is because efficient cryptographic constructions often lack of security proofs, and cryptographic constructions with strong security proofs are often not very efficient to be considered in practice.

This short introduction bespeaks that lattice problems are quite hard in a classical setting, but even for quantum computers these problems are hard. Until today, no efficient quantum algorithms are known. The first connection between lattice problems and quantum algorithms was done in [28], but no newsworthy efficient quantum algorithms came up from it. The existing quantum algorithms for lattice problems perform not much better than the existing classical ones. All knowledge on quantum computation seems to be not applicable in lattice problems, and therefore finding polynomial time quantum algorithms for lattice problems remains an open problem. Hereupon, it is believed that lattice-based cryptographic constructions are quantum-resistant.

3.1 Groundwork

A *lattice* is defined as the set of all integer combinations of n linearly independent vectors (over $\mathbb{R}$) $b_1, \ldots, b_n \in \mathbb{R}^n$. Given n linearly independent vectors (over $\mathbb{R}$) $b_1, \ldots, b_n \in \mathbb{R}^n$, we define the *lattice generated by* $b_1, \ldots, b_n$ as the set

$$L(b_1, \ldots, b_n) = \left\{ \sum_{i=1}^{n} x_i b_i : x_1, \ldots, x_n \in \mathbb{Z} \right\}.$$

The set of vectors $b_1, \ldots, b_n$ is called a *basis* of the lattice. A basis can be represented by a matrix $B = [b_1, \ldots, b_n] \in \mathbb{R}^{n \times n}$ having the basis vectors as columns. To ease notation, we write $L(B)$ for $L(b_1, \ldots, b_n)$ and say that $L(B)$ is the lattice generated by B. Hence $L(B) = \{Bx : x \in \mathbb{Z}^n\}$.

A lattice can have multiple bases, as enunciated by the next proposition. This fact is the essence of many cryptosystems, as we will evidence in section 3.3.

Proposition 1. *Let $L(B), L(B')$ be two lattices with $B, B' \in \mathbb{R}^{n \times n}$ as bases, respectively.*

1. *If U is a unimodular matrix, then U^{-1} is unimodular;*
2. *$L(B) = L(B')$ if and only if there exists a unimodular matrix U such that $B' = BU$.*

Proof. 1. Recall that U is said to be unimodular if it is an integer square matrix satisfying $\det(U) = \pm 1$. Since U is unimodular, $U \in \mathbb{Z}^{n \times n}$ and $\det(U) = \pm 1$. Then U is invertible and $\det(U^{-1}) = \det(U)^{-1} = \pm 1$. By the identity $U^{-1} = \det(U)^{-1} \cdot \mathrm{adj}(U)$ and since the entries of $\mathrm{adj}(U)$ are integers, we get $U^{-1} \in \mathbb{Z}^{n \times n}$.

2. Suppose that $L(B) = L(B')$ and let $B' = [b'_1, \ldots, b'_n]$. Then, $b'_1, \ldots, b'_n \in L(B') = L(B)$. Hence there exists $U \in \mathbb{Z}^{n\times n}$ such that $B' = BU$. By the same way, there exists $V \in \mathbb{Z}^{n\times n}$ such that $B = B'V$. Therefore $B' = BU = B'VU$. Taking determinants, we get $\det(B') = \det(B') \cdot \det(VU)$. Thus $\det(VU) = 1$ and we get $\det(U) = \pm 1$.
Suppose now that there exists a unimodular matrix U such that $B' = BU$. Writing $B' = [b'_1, \ldots, b'_n]$, we have $b'_1, \ldots, b'_n \in L(B)$ since U is an integer matrix. Therefore $L(B') \subseteq L(B)$. Since $B = B'U^{-1}$ and U^{-1} is unimodular, by the same way we get $L(B) \subseteq L(B')$.

□

The *determinant* of a lattice L is defined as $\det(L) = |\det(B)|$, where B is a basis of L. The value of the determinant is independent of the choice of the basis. To see this, we consider another basis B' of L, write $B' = BU$ where U is unimodular, and find that $\det(L) = |\det(B')| = |\det(BU)| = |\det(B) \cdot \det(U)| = |\pm \det(B)| = |\det(B)|$. Geometrically, this value corresponds to the inverse of the density of the lattice points in $\mathbb{R}^n$.

The *dual* of a lattice $L \subseteq \mathbb{R}^n$, denoted L^*, is given by

$$L^* = \{y \in \mathbb{R}^n : \langle x, y\rangle \in \mathbb{Z} \text{ for all } x \in L\}.$$

In fact, for this definition to make sense, we have to prove that L^* is a lattice.

Proposition 2. *If B is a basis of a lattice $L \subseteq \mathbb{R}^n$, then $L(B)^* = L((B^{-1})^T)$. Moreover, $(L^*)^* = L$ and $\det(L^*) = \det(L)^{-1}$ for any lattice L.*

Proof. Let $z \in L((B^{-1})^T)$. Then there exists $y \in \mathbb{Z}^n$ such that $z = (B^{-1})^T y$. Let $z' \in L(B)$. Then there exists $x \in \mathbb{Z}^n$ such that $z' = Bx$. Hence

$$\langle z', z\rangle = (z')^T z = (Bx)^T (B^{-1})^T y = x^T B^T (B^T)^{-1} y = x^T y \in \mathbb{Z}.$$

Therefore $z \in L(B)^*$ and $L((B^{-1})^T) \subseteq L(B)^*$.
Let $z \in L(B)^*$. Then $B^T z \in \mathbb{Z}^n$: writing $z = (z_1, \ldots, z_n) \in \mathbb{R}^n$, $B = [b_1, \ldots, b_n]$ and $b_i = (b_{i,1}, \ldots, b_{i,n})$ for each $i = 1, \ldots, n$, we have

$$B^T z = \begin{pmatrix} b_{1,1} & \cdots & b_{1,n} \\ \vdots & \ddots & \vdots \\ b_{n,1} & \cdots & b_{n,n} \end{pmatrix} \begin{pmatrix} z_1 \\ \vdots \\ z_n \end{pmatrix} = \begin{pmatrix} \begin{pmatrix} b_{1,1} & \cdots & b_{1,n} \end{pmatrix} \begin{pmatrix} z_1 \\ \vdots \\ z_n \end{pmatrix} \\ \vdots \\ \begin{pmatrix} b_{n,1} & \cdots & b_{n,n} \end{pmatrix} \begin{pmatrix} z_1 \\ \vdots \\ z_n \end{pmatrix} \end{pmatrix} = \begin{pmatrix} \langle b_1, z\rangle \\ \vdots \\ \langle b_n, z\rangle \end{pmatrix} \in \mathbb{Z}^n.$$

Hence $z = (B^{-1})^T B^T z \in L((B^{-1})^T)$ and $L(B)^* \subseteq L((B^{-1})^T)$.
Let $L = L(B)$ be any lattice. Then,

$$\det(L^*) = \det(L(B)^*) = \det(L((B^{-1})^T) = |\det((B^{-1})^T)| = |\det(B)|^{-1} = 1/\det(L).$$

Finally, $(L^*)^* = (L(B)^*)^* = L((B^{-1})^T))^* = L(((B^{-1})^T)^{-1})^T) = L(B) = L$. □

Example 1. Using the definition of dual lattice or using the previous proposition, we have $(t\mathbb{Z}^n)^* = \frac{1}{t}\mathbb{Z}^n$ for every $t \in \mathbb{R} \setminus \{0\}$.

If $b_1, \ldots, b_m \in \mathbb{R}^n$ are m linearly independent vectors (over $\mathbb{R}$), we can actually define the lattice generated by $b_1, \ldots, b_m$ as above. If L is the lattice generated by $b_1, \ldots, b_m$ and B is a matrix that represents the basis $b_1, \ldots, b_m$, the determinant of L is defined as $\det(L) = \sqrt{|\det(B^T B)|}$ and $L^* = \{y \in \text{span}_{\mathbb{R}}\{b_1, \ldots, b_m\} : \langle x, y\rangle \in \mathbb{Z} \text{ for all } x \in L\}$. These definitions generalize the previous ones. In addition, $L(B)^* = L(B(B^T B)^{-1})$. The previous results hold with the necessary modifications and their proofs are similar.

A lattice L is said to be *q-ary* if $q\mathbb{Z}^n \subseteq L \subseteq \mathbb{Z}^n$ for some integer q. If q is an integer multiple of $\det(L)$, then L is a q-ary lattice: letting $L = L(B)$, $q = k \cdot \det(L)$ for some $k \in \mathbb{Z}$ and $y \in q\mathbb{Z}^n$, we have $y = qx$ for some $x \in \mathbb{Z}^n$ and $B^{-1}y = B^{-1}qx = \pm k \cdot \text{adj}(B)x \in \mathbb{Z}^n$, and thus $y = B(\pm k \cdot \text{adj}(B)x) \in L(B) = L$. This proves that any integer lattice $L \subseteq \mathbb{Z}^n$ is a q-ary lattice for some integer q.

Given a matrix $B \in \mathbb{Z}_q^{n \times m}$ where $q, m, n \in \mathbb{Z}$, we define two m-dimensional q-ary lattices

$$L_q(B) = \{y \in \mathbb{Z}^m : y = Bx \pmod q \text{ for some } x \in \mathbb{Z}^n\}$$

and

$$L_q^{\perp}(B) = \{y \in \mathbb{Z}^m : By \equiv 0 \pmod q\}.$$

The orthogonal symbol for this second lattice is because this lattice is formed by the orthogonal vectors to the rows of B modulo q. This second lattice is called an *Ajtai lattice*. It can be shown that $L_q^{\perp}(B) = q \cdot L_q(B^T)^*$ and $L_q(B) = q \cdot L_q^{\perp}(B^T)^*$. Given this result, we say that these two lattices are dual to each other up to normalization.

Our main interest will be to find short vectors in random lattices, which is a well known computational problem on lattices. There are more computational problems on lattices, of which we have listed the most important ones below:

- The Shortest Vector Problem (SVP): Given a basis B of a lattice, find the shortest nonzero vector in $L(B)$;
- The Closest Vector Problem (CVP): Given a basis B of a lattice and a vector y, not necessarily in the lattice, find the lattice point $x \in L(B)$ closest to y. This is of course a generalization of the SVP problem;
- The Shortest Independent Vector Problem (SIVP): Given a basis $B \in \mathbb{Z}^{n \times n}$ of a lattice, find n linearly independent vectors $x_1, \ldots, x_n \in L(B)$ minimizing the quantity $\max_{i=1,\ldots,n} ||x_i||$.

Since sometimes it is difficult to solve these problems (for instance, when the dimension of the lattice is sufficiently large), one can consider the approximation variant of these problems, denoted with an additional subscript γ which indicates the approximation factor. For example, in SVP_γ the target is a vector whose norm is at most γ times the norm of the shortest nonzero vector.

In order to find such short vectors in random lattices we use Minkowski's Theorem applied with the Lebesgue's measure and $V = \mathbb{R}^n$:

Theorem 3 (Minkowski)**.** *Let μ be a Haar measure, V an euclidean space, K a limited convex symmetric by the origin of V and L a lattice in V. If $\mu(K) > 2^n\mu(V/L)$, then there exists $x \in L \setminus \{0\}$ such that $x \in K$. In other words, if $\mu(K) > 2^n\mu(V/L)$, then $L \cap K \neq \{0\}$.*

Remark 1. The proof of Minkowski's Theorem can be widely found in literature, such as in [16]. If μ is a translation invariant measure, for all $\mathbb{Z}$-basis $\{b_1, \ldots, b_n\}$ of L, the measure of the parallelepiped $P = \{\sum_{i=1}^n x_i b_i : 0 \leq x_i < 1\}$ is independent of the chosen basis and this value is called the *covolume* $\mu(V/L)$ of L. For $V = \mathbb{R}^n$ and the Lebesgue measure μ, the covolume of L is $\det(L)$.

Suppose that K is closed. For all $i = 1, \ldots, n$, let λ_i be the least real number such that $L \cap \lambda_i K$ contains i linearly independent over $\mathbb{Z}$ elements. These λ_i are called the *successive minima* of K on L. Minkowski's Theorem is equivalent to the say that $\lambda_1^n \mu(K) \leq 2^n \mu(V/L)$. Generally, we have for the successive minima:

Theorem 4 (Minkowski's 2nd Theorem)**.** *In the same conditions of Minkowski's Theorem above,*

$$\frac{2^n}{n!}\mu(V/L) \leq \lambda_1 \cdots \lambda_n \mu(K) \leq 2^n \mu(V/L).$$

Minkowski's Theorem can be used to give some interesting and well known results in number theory, such as: all prime numbers $p \equiv 1 \pmod 4$ are a sum of two squares; all positive integers are a sum of 4 squares.

Now suppose that K is the n-dimensional ball with radius 1, with the Lebesgue measure μ, in such a way that

$$\mu(K) = \frac{\pi^{\frac{n}{2}}}{\Gamma(1 + \frac{n}{2})}.$$

Then Minkowski's Theorem yields that

$$\lambda_1 \leq \frac{2}{\sqrt{\pi}} \Gamma\left(1 + \frac{n}{2}\right)^{\frac{1}{n}} \det(L)^{\frac{1}{n}}.$$

That is, the square of the euclidean norm of the smallest nonzero element of L is majorated by $\gamma_n \det(L)^{\frac{2}{n}}$, where γ_n is at most $\frac{4}{\pi}\Gamma\left(1 + \frac{n}{2}\right)^{\frac{2}{n}}$.
The best possible value for γ_n (valid for all lattices in $\mathbb{R}^n$) is called the n^{th} *Hermite constant*. Most of the values of γ_n are not currently known, i.e., finding the values of γ_n for all $n \in \mathbb{N}$ is an open problem. We can also prove that $\gamma_n \leq \left(\frac{4}{3}\right)^{\frac{n-1}{2}}$ (known as *Hermite's majoration*). The following table summarizes the known values of γ_n:

n	1	2	3	4	5	6	7	8	24
γ_n^n	1	$\frac{4}{3}$	2	4	8	$\frac{64}{3}$	64	256	4^{24}

From the geometrical interpretation of the determinant of a lattice, we can heuristically estimate λ_1 as the smallest radius of a ball whose volume is $\det(L)$. Using the formula of the volume of a n-dimensional ball of radius R

$$V_n(R) = \frac{\pi^{\frac{n}{2}}}{\Gamma\left(1+\frac{n}{2}\right)} R^n,$$

we get

$$\lambda_1 \approx \frac{1}{\sqrt{\pi}} \Gamma\left(1+\frac{n}{2}\right)^{\frac{1}{n}} \det(L)^{\frac{1}{n}} \approx (\pi n)^{\frac{1}{2n}} \cdot \sqrt{\frac{n}{2\pi e}} \cdot \det(L)^{\frac{1}{n}} \approx \sqrt{\frac{n}{2\pi e}} \cdot \det(L)^{\frac{1}{n}},$$

where we used Stirling's approximation and the behavior of $(\pi n)^{\frac{1}{2n}}$. From now on, for each lattice $L \subseteq \mathbb{R}^n$ we will denote by $\mathrm{GH}(L)$ and call *Gaussian Heuristic of* L the value

$$\mathrm{GH}(L) = \frac{1}{\sqrt{\pi}} \Gamma\left(1+\frac{n}{2}\right)^{\frac{1}{n}} \det(L)^{\frac{1}{n}}.$$

This heuristic is also valid for lattices in $\mathbb{R}^n$ of dimension $m \leq n$. In that case, we have

$$\mathrm{GH}(L) = \frac{1}{\sqrt{\pi}} \Gamma\left(1+\frac{m}{2}\right)^{\frac{1}{m}} \det(L)^{\frac{1}{m}}.$$

Let us observe what we can find for an Ajtai lattice $L_q^{\perp}(B)$, where $B \in \mathbb{Z}_q^{n \times m}$ is a random matrix such that $m \geq n$ and q is prime. Finding a short vector in $L_q^{\perp}(B)$ is equivalent to find a short solution for a set of n random equations modulo q in m variables. Since $m \geq n$ (and assuming that n is not too close to m), with high probability the rows of B are linearly independent over $\mathbb{Z}_q$. Thus the number of elements of $\mathbb{Z}_q^m$ which belong to $L_q^{\perp}(B)$ is exactly q^{m-n}. To see this, remark that the linear map $y \mapsto By$ has kernel of dimension $m-n$. Therefore we have $\det(L_q^{\perp}(B)) = q^n$. Applying the Gaussian heuristic given above, we get

$$\lambda_1(L_q^{\perp}(B)) \approx q^{\frac{n}{m}} \cdot \sqrt{\frac{m}{2\pi e}},$$

Next we introduce a technique for solving the Closest Vector Problem, called *Babai's Closest Vertex Algorithm* or *Babai's Rounding Technique*, which is pretty intuitive.

Theorem 5 (Babai's Closest Vertex Algorithm)**.** *Let $L \subseteq \mathbb{R}^n$ be a lattice with basis $b_1, \ldots, b_n$. If the basis vectors are sufficiently short and pairwise orthogonal, then the following algorithm solves CVP:*

Algorithm 1: Babai's Closest Vertex Algorithm / Babai's Rounding Technique

Input: A target vector $v = t_1 b_1 + \cdots + t_n b_n \in \mathbb{R}^n$

Output: A vector $b = a_1 b_1 + \cdots + a_n b_n \in L$

for $i = 1$ to n **do**

  $a_i \leftarrow \lfloor t_i \rceil$

end for

Output $b = a_1 b_1 + \cdots + a_n b_n$

Moreover, if the vectors of the basis are reasonably pairwise orthogonal, then the algorithm solves CVP_γ for some γ.

Remark 2. It is important to emphasize that Babai's algorithm is not useful if the basis vectors are highly non-orthogonal or if the basis vectors are not sufficiently short. In this case, Babai's algorithm generally returns a vector that is not in the lattice, normally far from the desired lattice point. When we have a basis of a lattice formed by short and almost orthogonal vectors, the Babai's Closest Vertex Algorithm permits to solve certain instances of the CVP, namely when the target vector is sufficiently close to the lattice.

3.2 Lattice reduction algorithms

In order to find better bases for lattices, i.e., bases consisting of short and nearly orthogonal vectors, it is imperative to consider algorithms capable of doing such task. Such algorithms are called *lattice reduction algorithms*, and they can be approximative or exact, as said in the introduction of this chapter.

We start with the LLL algorithm which is the most known lattice reduction algorithm, which dates back to 1982. Next we present the BKZ algorithm, which is an extension of the LLL algorithm. BKZ algorithm is the best on finding short vectors in a lattice, thus we will focus only on this algorithm and its improvements.

3.2.1 LLL algorithm

First we will recall some definitions and results of Linear Algebra. If V is a vector space and U is a subspace of V, we have $V = U \oplus U^\perp$ and therefore every element $v \in V$ can be written as $v = u + u'$, where $u \in U$ and $u' \in U^\perp$ are unique. We say that u is the orthogonal projection of v over U and write $u = \mathrm{pr}_U(v)$. Therefore $v = \mathrm{pr}_U(v) + \mathrm{pr}_{U^\perp}(v)$. If $\{u_1, \ldots, u_p\}$ is an orthogonal basis of U, for all $v \in V$ we have $\mathrm{pr}_U(v) = \frac{\langle v, u_1\rangle}{\langle u_1, u_1\rangle} u_1 + \cdots + \frac{\langle v, u_p\rangle}{\langle u_p, u_p\rangle} u_p$.

Let us recall the Gram-Schmidt orthogonalization process:

Proposition 3. *Let $\{b_1, \ldots, b_n\}$ be a basis of an euclidean space V. Define by recurrence the vectors*

- $b_1^* = b_1$;
- $b_i^* = b_i - \sum_{j=1}^{i-1} \mu_{i,j} b_j^*$, *where* $\mu_{i,j} = \frac{\langle b_i, b_j^*\rangle}{\langle b_j^*, b_j^*\rangle}$, *for all* $i = 2, \ldots, n$.

For all $i = 1, \ldots, n$, b_i^* is the orthogonal projection of b_i over the orthogonal of $\bigoplus_{j=1}^{i-1} \mathbb{R} b_j = \bigoplus_{j=1}^{i-1} \mathbb{R} b_j^*$ in $\bigoplus_{j=1}^{i} \mathbb{R} b_j$, and $\{b_1^*, \ldots, b_n^*\}$ is an orthogonal basis of V.

Proof. We have

$$\begin{aligned} \mathrm{pr}_{\left(\bigoplus_{j=1}^{i-1} \mathbb{R} b_j\right)^\perp}(b_i) &= \mathrm{pr}_{\left(\bigoplus_{j=1}^{i-1} \mathbb{R} b_j^*\right)^\perp}(b_i) \\ &= b_i - \mathrm{pr}_{\bigoplus_{j=1}^{i-1} \mathbb{R} b_j^*}(b_i) \end{aligned}$$

$$
\begin{aligned}
&= b_i - \frac{\langle b_i, b_1^* \rangle}{\langle b_1^*, b_1^* \rangle} b_1^* - \cdots - \frac{\langle b_i, b_{i-1}^* \rangle}{\langle b_{i-1}^*, b_{i-1}^* \rangle} b_{i-1}^* \\
&= b_i - \mu_{i,1} b_1^* - \cdots - \mu_{i,i-1} b_{i-1}^* \\
&= b_i - \sum_{j=1}^{i-1} \mu_{i,j} b_j^* \\
&= b_i^*.
\end{aligned}
$$

The second part of the proof can be found in any linear algebra book. □

Remark 3. If $\{b_1, \ldots, b_n\}$ is a basis of a lattice, $\{b_1^*, \ldots, b_n^*\}$ is not in general a basis of L.

Lemma 1. *Let L be a lattice and let $b_1, \ldots, b_n$ be a family of elements of L linearly independent over $\mathbb{Z}$. Let $B = [b_1 \cdots b_n]$ and $B^* = [b_1^* \cdots b_n^*]$. Then,* $\det(B) = \det(B^*)$.

Proof. Let

$$
\mu = \begin{pmatrix}
1 & 0 & \cdots & 0 & 0 \\
\mu_{2,1} & 1 & \cdots & 0 & 0 \\
\mu_{3,1} & \mu_{3,2} & \ddots & 0 & 0 \\
\vdots & \vdots & \ddots & \vdots & \vdots \\
\mu_{n-1,1} & \mu_{n-1,2} & \cdots & 1 & 0 \\
\mu_{n,1} & \mu_{n,2} & \cdots & \mu_{n,n-1} & 1
\end{pmatrix}_{n \times n}
$$

be the matrix of Gram-Schmidt coefficients. We have $B = B^* \mu^T$, and thus $\det(B) = \det(B^* \mu^T) = \det(B^*) \cdot 1 = \det(B^*)$. □

Lemma 2. *Let $\{b_1, \ldots, b_n\}$ be a basis of an euclidean space V. For all $i = 1, \ldots, n$, $||b_i^*|| \leq ||b_i||$.*

Proof. We have $b_i^* = b_i - \sum_{j=1}^{i-1} \mu_{i,j} b_j^*$, so $b_i = b_i^* + \sum_{j=1}^{i-1} \mu_{i,j} b_j^*$.
Then $||b_i||^2 = ||b_i^* + \sum_{j=1}^{i-1} \mu_{i,j} b_j^*||$.
Therefore

$$
\begin{aligned}
||b_i||^2 &= \left\langle b_i^* + \sum_{j=1}^{i-1} \mu_{i,j} b_j^*, b_i^* + \sum_{j=1}^{i-1} \mu_{i,j} b_j^* \right\rangle \\
&= ||b_i^*||^2 + 0 + 0 + \left\langle \sum_{j=1}^{i-1} \mu_{i,j} b_j^*, \sum_{j=1}^{i-1} \mu_{i,j} b_j^* \right\rangle \\
&= ||b_i^*||^2 + \underbrace{\sum_{j=1}^{i-1} \mu_{i,j}^2 ||b_j^*||^2}_{\geq 0},
\end{aligned}
$$

and now the result is clear. □

The following corollary provides, like Minkowski's 2nd Theorem, a lower bound of the product of the successive minima:

Corollary 1 (Hadamard)**.** *For all lattice L of V and all family $B = \{b_1, \ldots, b_n\}$ of elements of L linearly independent over $\mathbb{Z}$,*

1. $\det(L) = \prod_{i=1}^{n} ||b_i^*||$;
2. $\det(L) \leq \prod_{i=1}^{n} ||b_i||$;
3. $\det(L) = \prod_{i=1}^{n} ||b_i||$ *if and only if B is a basis of L and an orthogonal basis of V.*

Proof. By Lemma 1 and its proof, we have

$$\begin{aligned}
\det(L)^2 &= |\det(B)|^2 \\
&= \det(B)^2 \\
&= \det(B^T B) \\
&= \det\left((B^* M^T)^T B^* M^T\right) \\
&= \det(M(B^*)^T B^* M^T) \\
&= \det((B^*)^T B^*) \\
&= \det(\mathrm{diag}(||b_1^*||^2, \ldots, ||b_n^*||^2)) \\
&= \prod_{i=1}^{n} ||b_i^*||^2.
\end{aligned}$$

Then,

$$\det(L) = \prod_{i=1}^{n} ||b_i^*||$$

and the first part is proved.
By Lemma 2, we conclude that

$$\det(L) \leq \prod_{i=1}^{n} ||b_i||$$

and the second part is also proved.
If B is a basis of L and an orthogonal basis of V, we have $b_1^* = b_1, \ldots, b_n^* = b_n$ and by the previous part we have the desired equality.
Suppose now that $\det(L) = \prod_{i=1}^{n} ||b_i||$. Let $A = \{a_1, \ldots, a_n\}$ be a basis of L. Since $\{b_1, \ldots, b_n\}$ is linearly independent over $\mathbb{Z}$, there exists $M \in \mathbb{Z}^{n \times n}$ such that

$$[b_1 \cdots b_n] = [a_1 \cdots a_n] M$$

and $\det(M) \neq 0$. Therefore $\det(B) \neq 0$, which proves that $\{b_1, \ldots, b_n\}$ is linearly independent over $\mathbb{R}$. By Lemma 1 and Lemma 2 we get

$$\det(B)^2 = \det(B^*)^2 = \prod_{i=1}^{n} ||b_i^*||^2 \leq \prod_{i=1}^{n} ||b_i||^2.$$

We also have

$$\det(B)^2 = \det(A)^2 \det(M)^2 = \det(L)^2 \det(M)^2 = \left(\prod_{i=1}^{n} ||b_i||^2\right) \det(M)^2.$$

Thus, $\det(M)^2 = 1$, so $\det(M) = \pm 1$, which proves that $\{b_1, \ldots, b_n\}$ generates L. Hence $\{b_1, \ldots, b_n\}$ is a basis of L. We now have

$$\det(L) = \prod_{i=1}^{n} ||b_i^*|| = \prod_{i=1}^{n} ||b_i||.$$

By the proof of Lemma 2, $\mu_{i,j} = 0$ for all $1 \leq j < i \leq n$, which implies that $\langle b_i, b_j \rangle = 0$ for all $i \neq j$. Hence $\{b_1, \ldots, b_n\}$ is an orthogonal basis of L. □

The second part of the corollary proves that $\lambda_1 \cdots \lambda_n \geq \det(L)$ and suggests that the bases of L formed by short vectors are almost orthogonal. In fact, if $\{b_1, \ldots, b_n\}$ is a basis of L formed by short vectors, $\prod_{i=1}^{n} ||b_i||$ is close to $\det(L)$ and, by the third part of the corollary $\{b_1, \ldots, b_n\}$ is almost orthogonal.

This is what gives rise to the following definition. Let L be a lattice. We say that a basis $\{b_1, \ldots, b_n\}$ of L is LLL-reduced if, with Gram-Schmidt orthogonalization's notation,

- $|\mu_{i,j}| \leq \frac{1}{2}$ for all $1 \leq j < i \leq n$;
- $||b_i^* + \mu_{i,i-1} b_{i-1}^*||^2 \geq \frac{3}{4} ||b_{i-1}^*||^2$ for all $i = 2, \ldots, n$,
  or equivalently $||b_i^*||^2 \geq \left(\frac{3}{4} - \mu_{i,i-1}^2\right) ||b_{i-1}^*||^2$ for all $i = 2, \ldots, n$.

Remark 4.

- Notice that the vectors $b_i^* + \mu_{i,i-1} b_{i-1}^*$ and b_{i-1}^* are the orthogonal projections of b_i and b_{i-1}^* over the orthogonal of $\bigoplus_{j=1}^{i-2} \mathbb{R} b_j$.
  This can be seen by doing the same type of calculus done in Proposition 3.
- The constant $\frac{3}{4}$ in second inequality may be replaced by any fixed real number $\frac{1}{4} < \delta < 1$, and all of the following results can be adapted.
  When $\delta \neq \frac{3}{4}$, we shall refer to a LLL-δ reduction.
- From the second property, $||b_i^*||^2 \geq \left(\frac{3}{4} - \mu_{i,i-1}^2\right) ||b_{i-1}^*||$, and by the first property $\frac{1}{2} \leq \frac{3}{4} - \mu_{i,i-1}^2 \leq \frac{3}{4}$.
  Then the second property says that b_i^* is not much shorter than b_{i-1}^*.
- In the second property, we say that $||b_i^* + \mu_{i,i-1} b_{i-1}^*||^2 \geq \frac{3}{4} ||b_{i-1}^*||^2$ for all $i = 2, \ldots, n$ is equivalent to say that $||b_i^*||^2 \geq \left(\frac{3}{4} - \mu_{i,i-1}^2\right) ||b_{i-1}^*||^2$ for all $i = 2, \ldots, n$.
  This is because

$$\begin{aligned}
||b_i^*||^2 \geq \left(\frac{3}{4} - \mu_{i,i-1}^2\right) ||b_{i-1}^*||^2 &\iff ||b_i^*||^2 \geq \frac{3}{4} ||b_{i-1}^*||^2 - \mu_{i,i-1}^2 ||b_{i-1}^*||^2 \\
&\iff ||b_i^*||^2 + \mu_{i,i-1}^2 ||b_{i-1}^*||^2 \geq \frac{3}{4} ||b_{i-1}^*||^2 \\
&\iff \left\langle b_i^* + \mu_{i,i-1} b_{i-1}^*, b_i^* + \mu_{i,i-1} b_{i-1}^* \right\rangle \geq \frac{3}{4} ||b_{i-1}^*||^2 \\
&\iff ||b_i^* + \mu_{i,i-1} b_{i-1}^*||^2 \geq \frac{3}{4} ||b_{i-1}^*||^2.
\end{aligned}$$

Theorem 6. *Let $\{b_1, \ldots, b_n\}$ be a LLL-reduced basis of a lattice L of $\mathbb{R}^n$. Then,*

1. *For all $1 \leq j \leq i \leq n$, $||b_j|| \leq 2^{\frac{i-1}{2}}||b_i^*||$;*
2. *$\det(L) \leq \prod_{i=1}^{n} ||b_i|| \leq 2^{\frac{n(n-1)}{4}} \det(L)$;*
3. *Besides, $||b_1|| \leq 2^{\frac{n-1}{4}} \det(L)^{\frac{1}{n}}$;*
4. *For all $x \in L \setminus \{0\}$, $||b_1|| \leq 2^{\frac{n-1}{2}}||x||$.*
   More generally, for all $\{x_1, \ldots, x_t\}$ of linearly independent elements of L over $\mathbb{Z}$ and all $j \leq t$, we have $||b_j|| \leq 2^{\frac{n-1}{2}} \max\{||x_1||, \ldots, ||x_t||\}$.

Proof. 1. Let $1 \leq j < i \leq n$. We have

$$||b_i^*||^2 \geq \left(\frac{3}{4} - \mu_{i,i-1}^2\right) ||b_{i-1}^*||^2 \geq \frac{1}{2}||b_{i-1}^*||^2.$$

Then,

$$||b_1^*||^2 \leq 2||b_2^*||^2 \leq 2^2||b_3^*||^2 \leq \cdots \leq 2^{n-1}||b_n^*||^2,$$

i.e.,

$$||b_j^*||^2 \leq 2^{i-j}||b_i^*||^2.$$

Since

$$\begin{aligned} ||b_i||^2 &= ||b_i^*||^2 + \sum_{j=1}^{i-1} \mu_{i,j}^2 ||b_j^*||^2 \\ &\leq ||b_i^*||^2 + \sum_{j=1}^{i-1} \frac{1}{4} 2^{i-j} ||b_i^*||^2 \\ &= \left(1 + \frac{1}{4}(2^i - 2)\right) ||b_i^*||^2 \\ &\leq 2^{i-1}||b_i^*||^2, \end{aligned}$$

we have

$$||b_j||^2 \leq 2^{j-1}||b_j^*||^2 \leq 2^{j-1}2^{i-j}||b_i^*||^2 = 2^{i-1}||b_i^*||^2.$$

2. By Hadamard's Corollary we have $\det(L) = \prod_{i=1}^{n} ||b_i^*||$. Since $||b_i^*|| \leq ||b_i||$ and, by the first part, $||b_i|| \leq 2^{\frac{i-1}{2}}||b_i^*||$ for all $1 \leq i \leq n$, we get

$$\begin{aligned} \det(L) &= \prod_{i=1}^{n} ||b_i^*|| \\ &\leq \prod_{i=1}^{n} ||b_i|| \\ &\leq \prod_{i=1}^{n} 2^{\frac{i-1}{2}} ||b_i^*|| \end{aligned}$$

$$= \left(\prod_{i=1}^{n} 2^{\frac{i-1}{2}} \right) \det(L)$$
$$= 2^{\frac{n(n-1)}{4}} \det(L).$$

3. By the second part, we have

$$||b_1||^n = ||b_1|| \cdots ||b_1|| \leq 2^{\frac{1-1}{2}} ||b_1^*|| \cdots 2^{\frac{n-1}{2}} ||b_n^*|| = 2^{\frac{n(n-1)}{4}} \det(L).$$

4. Write $x = \sum_{i=1}^{n} c_i b_i = \sum_{i=1}^{n} \alpha_i b_i^*$, where $c_i \in \mathbb{Z}$ and $\alpha_i \in \mathbb{R}$. Since $x \neq 0$, there exists a maximum $k \in \{1, \ldots, n\}$ such that $c_k \neq 0$. By the definition of the Gram-Schmidt orthogonal vectors, we have $c_k = \alpha_k$. Since α_k is integer,

$$||x||^2 \geq \alpha_k^2 ||b_k^*||^2 \geq ||b_k^*||^2.$$

By the first part, $||b_1||^2 \leq 2^{i-1} ||b_i^*||^2 \leq 2^{n-1} ||b_i^*||^2$, hence

$$||b_1||^2 \leq 2^{n-1} ||b_k^*||^2 \leq 2^{n-1} ||x||^2$$

and the first piece of the part is proved.
Write $x_j = \sum_{i=1}^{n} c_{i,j} b_i$ where $c_{i,j} \in \mathbb{Z}$. For a fixed j, denote by i_j the largest i such that $c_{i,j} \neq 0$. By the first piece of this part, $||x_j||^2 \geq ||b_{i_j}^*||^2$. Renumber the x_j so that $i_1 \leq \cdots \leq i_t$. We now claim that $j \leq i_j$ for all $1 \leq j \leq t$. In order to get a contradiction, we suppose that $i_j < j$ for a certain j. We have

$$\begin{aligned}
x_1 &= c_{1,1} b_1 + \cdots + c_{n,1} b_n \\
&= c_{1,1} b_1 + \cdots + c_{i_1,1} b_{i_1} \quad \in \quad \mathbb{Z} b_1 + \cdots + \mathbb{Z} b_{i_1} \\
&\vdots \\
x_j &= c_{1,j} b_1 + \cdots + c_{n,j} b_n \\
&= c_{1,j} b_1 + \cdots + c_{i_j,j} b_{i_j} \quad \in \quad \mathbb{Z} b_1 + \cdots + \mathbb{Z} b_{i_j}
\end{aligned}$$

Then $x_1, \ldots, x_j \in \mathbb{Z} b_1 + \cdots + \mathbb{Z} b_{i_j}$. Since $i_j < j$, we obtain that $x_1, \ldots, x_j \in \mathbb{Z} b_1 + \cdots + \mathbb{Z} b_{j-1}$, which is a contradiction, given that $x_1, \ldots, x_j$ are linearly independent over $\mathbb{Z}$. Since $j \leq i_j$, we have, by the first part,

$$\begin{aligned}
||b_j||^2 &\leq 2^{i_j - 1} ||b_{i_j}^*||^2 \\
&\leq 2^{n-1} ||b_{i_j}^*||^2 \\
&\leq 2^{n-1} ||x_j||^2.
\end{aligned}$$

□

Next we describe the LLL algorithm, given in Algorithm 2, which transforms a given basis $\{b_1, \ldots, b_n\}$ of a lattice L into a LLL-reduced one.

If this algorithm ever terminates, we get a basis of L, since we only perform allowed column elementary operations on lattices, which are swapping two columns and adding a column to another one multiplied by an integer. Multiplying a column by an nonzero integer is not al-

Algorithm 2: LLL reduction algorithm

Input: A basis $\{b_1, \ldots, b_n\}$ of a lattice L
Output: A LLL-reduced basis for L

```
1  Start: Compute b_i^* and μ_{i,j} using Gram-Schmidt procedure
2  Reduction Step:
3    for i = 2 to n do
4      for j = i − 1 to 1, do
5        b_i ← b_i − ⌈μ_{i,j}⌋ b_j
6      end for
7    end for
8  Swap Step:
9    if ∃i such that (3/4)||b_i^*||^2 > ||μ_{i+1,i} b_i^* + b_{i+1}^*||^2 then
10     b_i ↔ b_{i+1}
11     goto 1
12   end if
```

lowed, unless this integer is ± 1.

Moreover, if the algorithm achieves completion, the second condition of a LLL-reduced basis is fulfilled. We now have to prove that the first condition is also verified, namely after the reduction step. As we only perform the allowed column elementary operations $b_i \leftarrow b_i - \lceil \mu_{i,j} \rfloor b_j$ where $i > j$, the Gram-Schmidt vectors and coefficients do not change. This is what the following lemma asserts.

Lemma 3. *Let $\{b_1, \ldots, b_n\}$ be a basis of an euclidean space V and let $\{b'_1, \ldots, b'_n\}$ be another basis of V such that $b'_i = b_i + \sum_{j<i} a_{i,j} b_j$, with $a_{i,j} \in \mathbb{Z}$. Then, the Gram-Schmidt vectors of these two basis are the same, i.e., $b_i^* = b_i'^*$ for all $1 \leq i \leq n$.*

Proof. This is done by induction. While doing the calculations keep in mind that $\mu_{i,i} = 1$ and $\mu_{i,j} = 0$ for $j > i$, as we have seen in the proof of Lemma 1. □

Since the Gram-Schmidt vectors do not change, after making the column operation $b_i \leftarrow b_i - \lceil \mu_{i,j} \rfloor b_j$ in the algorithm we have

$$\begin{aligned}
|\mu_{i,j}| &= \left| \frac{\langle b_i - \lceil \mu_{i,j} \rfloor b_j, b_j^* \rangle}{\langle b_j^*, b_j^* \rangle} \right| \\
&= \left| \frac{\langle b_i, b_j^* \rangle}{\langle b_j^*, b_j^* \rangle} - \lceil \frac{\langle b_i, b_j^* \rangle}{\langle b_j^*, b_j^* \rangle} \rfloor \cdot \frac{\langle b_j, b_j^* \rangle}{\langle b_j^*, b_j^* \rangle} \right| \\
&= \left| \frac{\langle b_i, b_j^* \rangle}{\langle b_j^*, b_j^* \rangle} - \lceil \frac{\langle b_i, b_j^* \rangle}{\langle b_j^*, b_j^* \rangle} \rfloor \cdot \mu_{j,j} \right| \\
&= \left| \frac{\langle b_i, b_j^* \rangle}{\langle b_j^*, b_j^* \rangle} - \lceil \frac{\langle b_i, b_j^* \rangle}{\langle b_j^*, b_j^* \rangle} \rfloor \right| \leq \frac{1}{2},
\end{aligned}$$

and this proves that after the reduction step the first condition of a LLL-reduced basis is satisfied.

Putting all together, if the algorithm ever terminates, its output is a LLL-reduced basis. In [21] and [29] we can find an analysis of the running time of the LLL algorithm. The strategy is to

bound the number of iterations and the running time of a single iteration. If L is a lattice with basis $b_1, \ldots, b_n$ and if $B \in \mathbb{R}_{\geq 2}$ is such that $|b_i|^2 \leq B$ for all $1 \leq i \leq n$, then the LLL algorithm has complexity $O(n^6(\log B)^3)$. If fast multiplication techniques are applied, it can be reduced to $O(n^{5+\epsilon}(\log B)^{2+\epsilon})$ for every $\epsilon > 0$.

It is worth mentioning that, after LLL-reduction, the first vector of the new obtained basis is relatively short. However, the first vector in the new basis obtained after LLL-reducing the basis is not necessarily the shortest one among the vectors of the new basis. If we want to explicit the shortest vector found by the LLL algorithm, we have to analyse the norm of the vectors of the obtained basis and extract the shortest one. Making δ close to 1 will improve the quality of the output basis, given the second property of the LLL-reduced bases.

Example 2. Consider the lattice L with basis

$$B = \begin{pmatrix} 16252 & 2404 & 41816 & 27419 & 62503 \\ 5541 & 46418 & 27475 & 53783 & 52719 \\ 48769 & 5335 & 56363 & 1412 & 22160 \\ 2593 & 59631 & 50417 & 42344 & 3940 \\ 22395 & 29390 & 62371 & 55983 & 64256 \end{pmatrix}.$$

The LLL-reduced basis (with $\delta = 0.999$) is given by

$$\begin{pmatrix} 1855 & 23160 & 29450 & 10069 & 4328 \\ 31849 & -24484 & -465 & -8429 & -17879 \\ -6182 & 2259 & 8167 & 24671 & 30280 \\ -5480 & -11807 & 2997 & -21117 & 29190 \\ 16007 & 10586 & -19059 & -18320 & 24708 \end{pmatrix}.$$

For this reduction we used the simple LLL algorithm implemented in NTL Library [39]. A stronger version of LLL algorithm is also implemented in NTL Library, it is called LLL_FP (Floating Point) and the result for this basis is the same.
Name the vectors of the basis B by $b_1, \ldots, b_n$ and the vectors of the basis LLL(B) by $b'_1, \ldots, b'_n$. We can see that the norm of b'_1 is greater than the norm of b'_3.
However, neither of these vectors is the shortest one in the lattice. The shortest vector in L is

$$\begin{aligned} u &= (-19381, -7964, 16504, -24114, 739) \\ &= 2b_1 - b_2 - b_3 + 2b_4 - b_5 \\ &= -b'_3 + b'_4, \end{aligned}$$

computed with an exact algorithm that we will present in the next section, or with the BKZ algorithm (with parameter $\beta \geq 3$).

To finish, we give some applications of the LLL algorithm:

1. Factoring polynomials over $\mathbb{Z}$ or $\mathbb{Q}$. In fact, LLL algorithm came up with this purpose;
2. Finding the minimal polynomial of an algebraic number α, given a good enough approximation of α;

3. Integer Linear Programming. Using LLL algorithm, we can obtain a solution in polynomial time.

4. Other lattice problems aside from SVP, such as CVP approximate;

5. Cryptanalysis, for example, in special cases of RSA and lattice-based cryptosystems;

6. Finding integer relations: given $x_1, \ldots, x_n \in \mathbb{R} \setminus \{0\}$, LLL algorithm can often determine if they are linearly dependent over $\mathbb{Z}$ and then find a relation $a_1x_1 + \cdots + a_nx_n$ with $a_1, \ldots, a_n \in \mathbb{Z}$. For this, we consider for all integer $M > 0$ the positive definite quadratic form $Q_M(a_1, \ldots, a_n) = a_2^2 + \cdots + a_n^2 + M(a_1x_1 + \cdots + a_nx_n)^2$. If M is big enough, the elements $a = (a_1, \ldots, a_n) \in \mathbb{Z}^n$ such that $Q_M(a)$ is small enough are candidates to provide such relations. As an example, we have the Machin's formula $\arctan(1) - 4\arctan(1/5) + \arctan(1/239) = 0$.

7. Obtain a basis from a given set of linearly dependent lattice vectors, using the previous topic. This will be specially useful for BKZ reduction algorithm presented below.

3.2.2 BKZ algorithm

The Blockwise-Korkine-Zolotarev (BKZ) algorithm dates back from 1987 and was introduced by Schnorr and Euchner. Despite of being the best reduction algorithm known in practice, BKZ uses an exact reduction algorithm as a subroutine (more specifically an enumeration subroutine) which is usually super-exponential in time. This enumeration subroutine can be quickened with some pruning techniques [11]. The time complexity of the BKZ algorithm is essentially unknown, and for that reason it became necessary some algorithm permitting to predict the quality of the output basis and to estimate the necessary time for BKZ-reducing the basis, without the need to run BKZ itself. In 2011, Yuanmi Chen and Phong Nguyen published the called BKZ simulation algorithm, together with an improvement of the BKZ algorithm, BKZ 2.0.

To properly explain the BKZ algorithm, we introduce some new definitions. First we define the orthogonal projections

$$\begin{aligned} \mathrm{pr}_i \colon \mathbb{R}^n &\to \langle b_1, \ldots, b_{i-1} \rangle^{\perp} \\ b &\mapsto b - \mathrm{pr}_{\bigoplus_{j=1}^{i-1} \mathbb{R} b_j^*}(b) = b - \frac{\langle b, b_1^* \rangle}{\langle b_1^*, b_1^* \rangle} b_1^* - \cdots - \frac{\langle b, b_{i-1}^* \rangle}{\langle b_{i-1}^*, b_{i-1}^* \rangle} b_{i-1}^* \end{aligned}$$

for $1 \leq i \leq n$. Next, for $1 \leq j \leq k \leq n$, we define the *local (projected) lattice* $L_{[j,k]}$ as

$$L_{[j,k]} = \mathrm{span}_{\mathbb{Z}}\{\mathrm{pr}_j(b_j), \ldots, \mathrm{pr}_j(b_k)\}$$

and the *local block* $B_{[j,k]}$ as

$$B_{[j,k]} = \{\mathrm{pr}_j(b_j), \ldots, \mathrm{pr}_j(b_k)\}.$$

From these definitions, it is clear that $B_{[j,k]}$ is a basis of $L_{[j,k]}$ and $L_{[j,k]} = L(B_{[j,k]})$.

For our purposes, the considered local blocks are $B_{[i,\min\{i+\beta-1,n\}]}$, where β is an additional

parameter called *blocksize*. Explicitly the considered local blocks are

$$B_{[1,\beta]}, B_{[2,\beta+1]}, \ldots, B_{[t,\beta+t-1]}, B_{[t+1,n]}, \ldots, B_{[n,n]}$$

where $B_{[t,\beta+t-1]}$ is the last block of size β (meaning that $\beta + t - 1 = n$). The following blocks decrease successively in size by one unit. The BKZ algorithm aims to ensure that the first vector in each of these blocks is the shortest vector in the corresponding local projected lattice, i.e., in the lattice generated by the corresponding local block.

Because of that, the BKZ algorithm needs an exact algorithm for finding shortest vectors. As β increases, the output basis is more reduced, but the running time of the algorithm increases, since the local projected lattices get bigger in dimension. Since the running time of BKZ algorithm is influenced by the blocksize β and the dimension n of the lattice, some values for (n, β) may lead the algorithm to take too much time in the reduction. However, one can abort the algorithm after some hours or days hoping to get a good enough basis for the intended purpose. As said in [5], Hanrot *et al.* proved that the bases obtained by aborting BKZ are only slightly worse than the ones obtained after the completion of BKZ. One can use the implementation of BKZ algorithm in NTL Library to study the running time of BKZ according to both n and β.

The algorithm has the following inputs: the basis of a lattice, a blocksize $\beta \in \{2, \ldots, n\}$ and a factor δ used in the LLL algorithm. The main idea of BKZ algorithm is to make the output basis have the shortest Gram-Schmidt vectors possible. For that, the algorithm ensures that all local blocks $B_{[j,\min(j+\beta-1,n)]}$ have as first vector the shortest one in the corresponding local projected lattice. If this happens, the new b_j^* is the smallest vector in the local projected lattice $L_{[j,\min(j+\beta-1,n)]}$, since $b_j^* = \mathrm{pr}_j(b_j)$. The BKZ algorithm is described in Algorithm 3.

Algorithm 3: BKZ reduction algorithm

Input: A basis $\{b_1, \ldots, b_n\}$ of a lattice L, a blocksize $\beta \in \{2, \ldots, n\}$ and a factor δ for LLL

Output: A BKZ-β reduced basis of L

1 LLL($b_1, \ldots, b_n$,μ)
2 $z \leftarrow 0$
3 $j \leftarrow 0$
4 **while** $z < n - 1$ **do**
5 $\quad j \leftarrow (j \pmod{n-1}) + 1$; $\quad k \leftarrow \min\{j + \beta + 1, n\}$ *//define the local block*
6 $\quad h \leftarrow \min\{k + 1, n\}$
7 $\quad v \leftarrow$Enum($L_{[j,k]}$) *//find* $v = (v_j, \ldots, v_k) \in \mathbb{Z}^{k-j+1} \setminus \{0\}$ *such that* $||\mathrm{pr}_j(\sum_{i=j}^{k} v_i b_i)|| = \lambda_1(L_{[j,k]})$
8 $\quad$ **if** $v \neq (1, 0, \ldots, 0)$ **then**
9 $\quad\quad z \leftarrow 0$
10 $\quad\quad$ LLL($b_1, \ldots, b_{j-1}, \sum_{i=j}^{k} v_i b_i, b_j, \ldots, b_h$,$\mu$)
11 $\quad$ **else**
12 $\quad\quad z \leftarrow z + 1$
13 $\quad\quad$ LLL($b_1, \ldots, b_h$,μ)
14 $\quad$ **end if**
15 **end while**
16 Output $b_1, \ldots, b_n$

As we can see in Algorithm 3, we begin with preprocessing the given basis using the LLL algorithm. Before moving to the next block, we also LLL-reduce the basis $\{b_1, \ldots, b_h\}$, where $h = \min\{k+1, n\}$, for preprocessing the next block.

If the first vector of a block $B_{[j,k]}$ is the shortest one in the local projected lattice $L_{[j,k]}$, we LLL-reduce the basis $\{b_1, \ldots, b_h\}$ and we proceed to the next block $B_{[j+1,\min\{k+1,n\}]}$. If not, the Enum subroutine permits to find a linear combination of $b_{\text{new}} = v_j b_j + \cdots + v_k b_k$ such that $\text{pr}_j(b_{\text{new}})$ is the shortest vector in $L_{[j,k]}$. After that, we insert b_{new} in the current basis, between b_{j-1} and b_j, and LLL-reduce the resultant basis $b_1, \ldots, b_{j-1}, b_{\text{new}}, b_j, \ldots, b_h$ for solving the linear dependency. This application of LLL may spoil things for some previous local blocks. Because of that, we define j as it is described in the algorithm, to guarantee that after processing the last block we come back to the first block, and we also add a variable z in the algorithm with the function of controlling the termination of the algorithm. When $z = n$, which means that all local blocks have as first vector the shortest one in the local projected lattice, the algorithm terminates.

We remark that the notation $\text{LLL}(b_1, \ldots, b_n; \mu)$ refers to a LLL-reducing of the basis $\{b_1, \ldots, b_n\}$ and an updating of the Gram-Schmidt coefficients. We can also improve the efficiency if the algorithm by permitting LLL algorithm to begin at a certain stage, since we know that the basis $\{b_1, \ldots, b_{j-1}\}$ or $\{b_1, \ldots, b_{h-1}\}$, depending on the *if* clause of the algorithm, is already LLL-reduced. When we do this we have to make sure that if an insertion has occurred then we place the new vector b_{new} in the right place. It is also clear that the BKZ algorithm is stronger than LLL algorithm, the output basis of BKZ algorithm is at least LLL-reduced.

Enumeration Subroutine

To ease notation, avoiding local projected lattices and local blocks, we describe the algorithm of this subroutine for a general lattice. One can easily adapt the algorithm for the BKZ algorithm. The idea is to describe a basic enumeration algorithm that can output the shortest vector of an arbitrary lattice. More details and optimizations of this subroutine (applied to BKZ algorithm) can be found in [5] and [11].

Let us now explore the enumeration subroutine of the BKZ reduction algorithm. This is an exact algorithm to find the shortest vector in a lattice, and so as the dimension of the lattice increases, the running time of the algorithm grows exponentially. As the name suggests, the idea consists on enumerating all possible vectors in the lattice whose length is bounded by some constant that we will disclose below.

So let $L \subseteq \mathbb{R}^n$ be a lattice and let $B = [b_1, \ldots, b_m]$ be a matrix that represents a basis for this lattice. We begin by defining the search space of the enumeration as the set of all coefficient vectors $u \in \mathbb{Z}^m$ such that

$$\left\| \sum_{t=1}^{m} u_t b_t \right\|^2 \leq C,$$

where $C = ||b_1^*||^2 = ||b_1||^2$. This choice for such a constant seems very naive, but there is a reason for it. When the basis of the lattice is LLL-reduced, b_1 is probably the shortest known vector of the lattice. Recall that the enumeration subroutine is called after the basis has been LLL-reduced. While the algorithm search in the tree the shortest vector of the lattice, this bound

C is updated for efficiency reasons, permitting cutting some branches of the tree. This constant C can initially be lesser than $||b_1||^2$ if we know more or less the length of the shortest vector of the lattice. Feeding the algorithm a better value for C will reduce the search space of the algorithm, but of course if we attribute to C a too low value, the algorithm returns nothing useful.

Using Gram-Schmidt notation, we have

$$\begin{aligned}
\left\|\sum_{t=1}^{m} u_t b_t\right\|^2 &= \left\|\sum_{t=1}^{m} u_t \left(b_t^* + \sum_{j=1}^{t-1} \mu_{t,j} b_j^*\right)\right\|^2 \\
&= \left\|\sum_{t=1}^{m} u_t b_t^* + \sum_{t=1}^{m}\sum_{j=1}^{t-1} u_t \mu_{t,j} b_j^*\right\|^2 \\
&= \left\|\sum_{t=1}^{m} u_t b_t^* + \sum_{t=1}^{m} u_t \mu_{t,1} b_1^* + \cdots + \sum_{t=1}^{m} u_t \mu_{t,m} b_m^*\right\|^2 \\
&= \left\|\left(u_1 + \sum_{t=1}^{m} u_t \mu_{t,1}\right) b_1^* + \cdots + \left(u_m + \sum_{t=1}^{m} u_t \mu_{t,m}\right) b_m^*\right\|^2 \\
&= \left\|\left(u_1 + \sum_{t=1}^{m} u_t \mu_{t,1}\right) b_1^*\right\|^2 + \cdots + \left\|\left(u_m + \sum_{t=1}^{m} u_t \mu_{t,m}\right) b_m^*\right\|^2 \\
&= \left(u_1 + \sum_{t=1}^{m} u_t \mu_{t,1}\right)^2 ||b_1^*||^2 + \cdots + \left(u_m + \sum_{t=1}^{m} u_t \mu_{t,m}\right)^2 ||b_m^*||^2 \\
&= \sum_{k=1}^{m} \left(u_k + \sum_{t=1}^{m} u_t \mu_{t,k}\right)^2 ||b_k^*||^2 \\
&= \sum_{k=1}^{m} \left(u_k + \underbrace{\sum_{t=k+1}^{m} u_t \mu_{t,k}}_{c_k}\right)^2 ||b_k^*||^2 \\
&= \sum_{k=1}^{m} (u_k + c_k)^2 ||b_k^*||^2.
\end{aligned}$$

We now define

$$\begin{aligned}
l_m &= (u_m + c_m)^2 ||b_m^*||^2 \\
l_{m-1} &= l_m + (u_{m-1} + c_{m-1})^2 ||b_{m-1}^*||^2 \\
&\vdots \\
l_1 &= l_2 + (u_1 + c_1)^2 ||b_1^*||^2.
\end{aligned}$$

In general and by recursion, $l_k = l_{k+1} + (u_k + c_k)^2 ||b_k^*||^2$ for all $k = 1, \ldots, m-1$ and $l_m = (u_m + c_m)^2 ||b_m^*||^2$. The value of l_k corresponds to the last k parts of the above sum, which is $\|\sum_{t=k}^{n} u_t b_t\|$.

Now we can write the enumeration algorithm, Algorithm 4, which is given in [7]. In the algo-

rithm, t represents the level where we are in the tree. The following figure exemplifies the aspect of such trees. The leafs are located in level 1 and the root of the enumeration tree is a virtual vertex. If this vertex did not exist, instead of a tree we would have a forest.

Algorithm 4: A basic enumeration algorithm

Input: Gram-Schmidt coefficients $\mu_{i,j}$ and the squared norms of the Gram-Schmidt vectors $||b_1^*||^2, \ldots, ||b_m^*||^2$ of a lattice with basis $\{b_1, \ldots, b_m\}$

Output: A vector of coefficients $u = (u_1, \ldots, u_m) \in \mathbb{Z}^m$ such that $\sum_{t=1}^{m} u_t b_t$ is the shortest vector in the lattice

1 $C \leftarrow ||b_1^*||^2$
2 $u \leftarrow (1, 0, \ldots, 0) \in \mathbb{Z}^m$
3 $u_{\min} \leftarrow (1, 0, \ldots, 0) \in \mathbb{Z}^m$
4 $l \leftarrow (0, \ldots, 0) \in \mathbb{Z}^{m+1}$
5 $c \leftarrow (0, \ldots, 0) \in \mathbb{Z}^m$
6 $t = 1$
7 **while** $t < m - 1$ **do**
8 $l_t \leftarrow l_{t+1} + (u_t + c_t)^2 ||b_t^*||^2$
9 **if** $l_t < C$ **then**
10 **if** $t = 1$ **then**
11 $u_{\min} \leftarrow u$
12 $C \leftarrow l_t$
13 **else**
14 $t \leftarrow t - 1$
15 $c_t \leftarrow \sum_{i=t+1}^{m} u_i \mu_{i,t}$
16 $u_t \leftarrow -\lfloor c_t \rceil$
17 **end if**
18 **else**
19 $t \leftarrow t + 1$
20 Choose next u_t using the zig-zag pattern
21 **end if**
22 **end while**
23 Output $u_{\min}$

The starting point for this exhaustive search is the vector $u = (1, 0, \ldots, 0)$, since we defined the initial bound as $C = ||b_1^*|| = ||b_1||$. In the algorithm, $u_{\min}$ is the vector of coefficients which defines the shortest vector found. We initialize it as $u_{\min} = u = (1, 0, \ldots, 0)$. The vector l has coordinates l_k defined as before, so we must initialize it as $l = (||b_1||^2, 0, \ldots, 0)$. Since the algorithm computes l_1 at line 8, we actually can initialize l as $l = (0, \ldots, 0)$. This vector has to have an extra coordinate $l_{m+1} = 0$ because of the computation of line 8. Since we have initialized u as $u = (1, 0, \ldots, 0)$, we must initialize c, whose coordinates were defined before, as $c = (0, \ldots, 0)$.

Assuming that we are in level t of the tree, we can have $l_t < C$, which means that the current (squared) norm $\|\sum_{k=t}^{n} u_k b_k\|^2$ is lesser than the actual bound C, leading to two cases:

1. If $t = 1$, i.e., if we land on a new leaf, the bound C and $u_{\min}$ are both updated;

2. If $t \neq 1$, we move one level down in the tree, compute the coefficient c_t and set $u_t = -\lfloor c_t \rceil$, as this value is the best one for minimizing the (squared) norm $l_t = l_{t+1} + (u_t + c_t)^2 ||b_t^*||^2$. In [7] there is a mistake in this part of the algorithm which we corrected.

If $l_t \geq C$, we move one level up in the tree and choose the next u_t using the zig-zag pattern $-\lfloor c_t \rceil + 1, -\lfloor c_t \rceil - 1, -\lfloor c_t \rceil + 2, -\lfloor c_t \rceil - 2, \ldots$

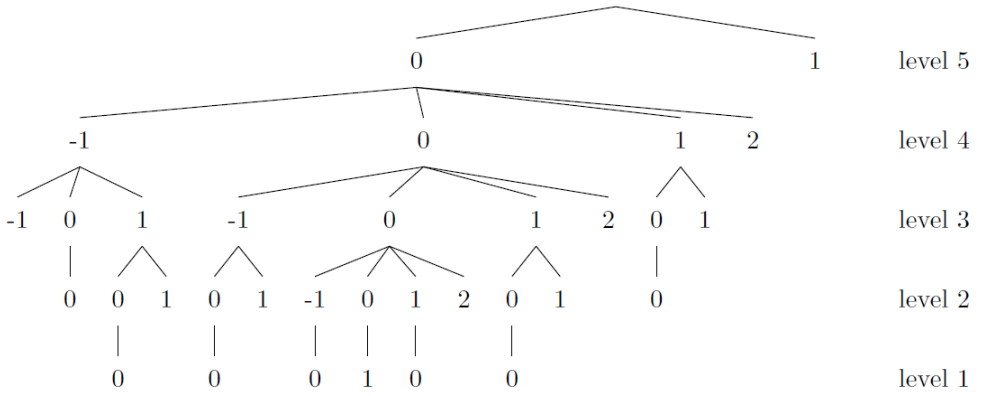

Fig. 3.1: Example of an enumeration tree.
This is the enumeration tree corresponding to the LLL-reduced basis of Example 2 in the section of LLL algorithm.

BKZ 2.0 algorithm

The BKZ 2.0 algorithm is an updated version of the original BKZ algorithm, given by Chen and Nguyen in [5]. This version includes four improvements, and one of them is simply an *early-abort* (which is common in cryptanalysis), that is, add a parameter that specifies how many iterations should be performed. We next give an idea of the other three improvements.

1. Due to the cost of the enumeration subroutine, it is useful to consider some pruning techniques to shorten the enumeration tree. BKZ 2.0 uses such pruning techniques, given in [11] by Gama, Nguyen and Regev, that provide asymptotical speedups in the running time of BKZ.

2. Before processing a local block, the corresponding basis is LLL-reduced. However, LLL-reducing the basis can worsen its quality. Therefore, Chen and Nguyen implemented in BKZ an additional function, which aims to ensure that bases are more than LLL-reduced before processing local blocks, without big time consumption.

3. The enumeration subroutine of BKZ algorithm uses the initial bound $C = ||b_j^*||^2$ when searching $\lambda_1(L_{[j,k]})$, and thus for decreasing the running time of it we may consider a lower initial bound. For this, we use the Gaussian Heuristic of $L_{[j,k]}$. However, when the dimension of the local projected lattices used in BKZ algorithm is low, say ≤ 30, those local projected lattices do not behave like random lattices and therefore the Gaussian Heuristic cannot be used. Hence, for all local project lattices except the last 30 ones, we choose as initial bound $C = \min\{||b_j^*||^2, \gamma^2 \mathrm{GH}(L_{[j,k]})\}$, where γ is typically 1.05.

The BKZ 2.0 algorithm was used for breaking some lattice records and for solving some SVP and Darmstadt's lattice challenges. The goal of SVP challenges [31] is to find a nonzero lattice vector of norm $\leq 1.05\mathrm{GH}(L)$, where L is a random integer lattice of large volume, whilst in Darmstadt's lattice challenges [22] the goal is to find a vector of norm $< q$ in an Ajtai lattice, where the modulus q depends on the dimension of the lattice (for example, in the *toy challenge* of dimension 200 found in [22], $q = 30$).

BKZ simulation algorithm

Considering the BKZ algorithm, we define a round in BKZ as a complete iteration of j from 1 to $n-1$. Given a lattice L with basis $\{b_1, \ldots, b_n\}$, the BKZ simulation algorithm predicts the norms $||b_1^*||, \ldots, ||b_n^*||$ after N rounds in BKZ. We will denote by $||b_1^{*\prime}||, \ldots, ||b_n^{*\prime}||$ the norms of the Gram-Schmidt vectors of the basis $\{b_1, \ldots, b_n\}$ after N rounds in BKZ.

For efficiency reasons, the BKZ simulation algorithm works with the logarithms of these norms. Hence, and to ease writing, we define for $1 \leq i \leq n$

$$l_i = \log(||b_i^*||) \quad \text{and} \quad l_i' = \log(||b_i^{*\prime}||).$$

The BKZ simulation algorithm has as inputs the norms l_i, the blocksize β used in BKZ and the number of rounds N in BKZ to simulate. It can be used for determining the necessary number of rounds in BKZ until a good basis is found.

In the algorithm, j denotes the current round and local blocks are denoted by $B_{[k,f]}$. Comparing with the BKZ algorithm, we changed the role of j and k, but we preferred to keep the notation of [5], although it is easy to change the simulation algorithm to match notation.

It is said that a basis of a lattice L is *HKZ-reduced* (Hermite-Korkine-Zolotarev reduction) if

- $|\mu_{i,j}| \leq \frac{1}{2}$ for all $1 \leq j < i \leq m$;
- $||b_i^*|| = \lambda_1(L_i)$ for all $1 \leq i \leq m$,

where $L_i = \mathrm{pr}_i(L)$. Since $L_1 = L$ (recall that pr_1 is the identity map), b_1 is the shortest vector of L.

Proposition 4. *We have:*

1. $L_i = \mathrm{pr}_i(L) = L_{[i,n]}$;
2. *After a round in BKZ, the last blocks $B_{[t,n]}, \ldots, B_{[n,n]}$ (where $t = n - \beta + 1$) are HKZ-reduced;*
3. *HKZ reduction is stronger than BKZ reduction.*

Each of these statements can be easily checked. We now have the following chain: BKZ reduction is stronger than LLL reduction (as we could see before) and HKZ reduction is stronger than BKZ reduction. We also remark that not all local blocks are HKZ-reduced after a round in BKZ, just the last ones.

In random lattices, the length of the shortest vector can be approximated using the Gaussian heuristic GH. The BKZ simulation algorithm uses this fact to predict the length of the shortes vector of local projected lattices used in the BKZ algorithm, assuming that those lattices behave like random lattices. As it was experienced by Gama and Nguyen in [5], non-extreme local blocks $L_{[j,j+\beta-1]}$ have a ratio λ_1/GH that converge to the ratio of a random lattice of dimension β as β increases. The intuitive reason for the approximation of these two ratios is that when the dimension increases, there are more and more random lattices in the set of all lattices. When $\beta \geq 45$, the approximation gets more and more evident. Therefore, the BKZ simulation algorithm requires a blocksize $\beta \geq 45$.

Therefore, we predict $\lambda_1(L_{[j,k]})$ as:

1. For non-extreme local blocks, i.e., for the local blocks $B_{[1,\beta]}, \ldots, B_{[n-\beta,n-1]}$, we use the Gaussian Heuristic $\mathrm{GH}(L_{[j,k]})$, unless $||\mathrm{pr}_j(b_j)|| = ||b_j^*||$ is already better.

2. For the extreme local blocks $B_{[n-\beta+1,n]}, \ldots, B_{[n,n]}$, since they are HKZ-reduced, we suppose that they behave like HKZ-reduced bases from random lattices of the same volume. Because of that, the ideal would be to compute, according with the chosen blocksize β, the average norm of the corresponding HKZ-reduced lattice. However, when β is too large, computing these averages may take too much time. Then,

   - We precompute these average norms for the last 45 local blocks $B_{[n-44,n]}, \ldots, B_{[n,n]}$.
     They are computed experimentally, where an example is given in [12].
   - We use the Gaussian Heuristic for the other $\beta - 45$ local blocks $B_{[n-\beta+1,n]}, \ldots, B_{[n-45,n]}$.

Now we can write the BKZ simulation algorithm, which is given in Algorithm 5.

As said before, k and f define the local block, and the variable d denotes the dimension of the local projected lattice. The variable c_d stores the logarithm of the volume of a d-dimensional ball of radius 1, used in the calculation of the Gaussian Heuristic. The variable ϕ is a boolean variable which stores *false* if $||b_k^*||$ is compared with $\mathrm{GH}(L_{[k,f]})$ and $\mathrm{GH}(L_{[k,f]})$ is better. This corresponds to the insertion of a new vector in the current basis in BKZ algorithm. After the insertion of a new vector in the current basis and the LLL-reducing in BKZ algorithm, i.e., after the change of the variable ϕ to *false*, the following Gram-Schmidt vectors $b_{k+1}^*, ..., b_{n-45}^*$ may not be in the local projected lattice anymore. Because of this, we use the Gaussian Heuristic for the next blocks, skipping the comparison of line 17.

We have

$$\log\left(\mathrm{GH}(L_{[k,f]})\right) = \log\left(\det(L_{[k,f]})^{\frac{1}{d}} \cdot \frac{\Gamma\left(\frac{d}{2}+1\right)^{\frac{1}{d}}}{\sqrt{\pi}}\right) = \frac{\log\left(\det(L_{[k,f]})\right)}{d} + c_d.$$

The following Lemma will allow us to compute the volume of each local projected lattice $L_{[k,f]}$, i.e., $\det(L_{[k,f]})$, stored in the variable V in the algorithm.

Lemma 4. *We have* $\det(L_{[1,k]}) = \det(L_{[1,j-1]}) \cdot \det(L_{[j,k]})$.

Proof. We know that pr_1 is the identity map, so $\mathrm{pr}_1(b_1) = b_1, \ldots, \mathrm{pr}_1(b_k) = b_k$. Then, $\{\mathrm{pr}_1(b_1), \ldots, \mathrm{pr}_1(b_k)\} = \{b_1, \ldots, b_k\}$ and, by Hadamard's Theorem,

$$\det(L_{[1,k]}) = \prod_{i=1}^{k} ||b_i^*||.$$

By the same reason,

$$\det(L_{[1,j-1]}) = \prod_{i=1}^{j-1} ||b_i^*||.$$

To finish, we calculate $\det(L_{[j,k]})$. In order to apply Hadamard's Theorem, we need to compute

Algorithm 5: BKZ Simulation Algorithm

Input: The logarithms of the Gram-Schmidt norms $l_i = \log(||b_i^*||)$, for $i \in \{1, \ldots, n\}$, the desired blocksize $\beta \in \{45, \ldots, n\}$ and the number of rounds N to simulate

Output: A prediction for the logarithms of the Gram-Schmidt norms $l'_i = \log(||b_i^{*\prime}||)$ after N rounds of BKZ reduction

for $k = 1$ to 45 **do**
  $r_k \leftarrow$ average $\log(||b_k^*||)$ of a HKZ-reduced random unit volume 45-dimensional lattice
end for
for $d = 46$ to β **do**
  $c_d \leftarrow \log(\frac{\Gamma(\frac{d}{2}+1)^{\frac{1}{d}}}{\sqrt{\pi}})$
end for
for $i = 1$ to n **do**
  $l'_i = l_i$
end for
for $j = 1$ to N **do**
  $\phi \leftarrow$ true
  for $k = 1$ to $n - 45$ **do**
    $d \leftarrow \min\{\beta, n - k + 1\}$
    $f \leftarrow \min\{k + \beta - 1, n\}$
    $\log(V) \leftarrow \sum_{i=1}^{f} l_i - \sum_{i=1}^{k-1} l'_i$
    if $\phi =$ true **then**
      if $\frac{\log(V)}{d} + c_d < l_k$ **then**
        $l'_k \leftarrow \frac{\log(V)}{d} + c_d$
        $\phi \leftarrow$ false
      end if
    else
      $l'_k \leftarrow \frac{\log(V)}{d} + c_d$
    end if
  end for
  $\log(V) \leftarrow \sum_{i=1}^{n} l_i - \sum_{i=1}^{n-45} l'_i$
  for $k = n - 44$ to n **do**
    $l'_k \leftarrow \frac{\log(V)}{45} + r_{k+45-n}$
  end for
  for $i = 1$ to n **do**
    $l_i \leftarrow l'_i$
  end for
end for

the Gram-Schmidt vectors $\mathrm{pr}_j(b_j)^*, \ldots, \mathrm{pr}_j(b_k)^*$. Using the equality

$$\mathrm{pr}_j(b_{j+p}) = b^*_{j+p} + \sum_{t=j}^{j+p-1} \frac{\langle b_{j+p}, b^*_t \rangle}{\langle b^*_t, b^*_t \rangle} b^*_t$$

for all $0 \leq p \leq k - j$, we get, by Gram-Schmidt orthogonalization, $\mathrm{pr}_j(b_j)^* = b^*_j$, $\mathrm{pr}_j(b_{j+1})^* = b^*_{j+1}$, $\ldots$, $\mathrm{pr}_j(b_k)^* = b^*_k$. Therefore

$$\det(L_{[j,k]}) = \prod_{i=j}^{k} ||\mathrm{pr}_j(b_i)^*|| = \prod_{i=j}^{k} ||b^*_i||.$$

□

By Lemma 4, we have

$$\det(L_{[k,f]}) = \frac{\det(L_{[1,f]})}{\det(L_{[1,k-1]})} = \frac{\prod_{i=1}^{f} ||b^*_i||}{\prod_{i=1}^{k-1} ||b^{*\prime}_i||} = \frac{\prod_{i=1}^{f} ||l_i||}{\prod_{i=1}^{k-1} ||l'_i||}.$$

since a change in the basis $\{b_1, \ldots, b_f\}$ does not change the volume of $L_{[1,f]}$. Taking logarithms, we get $\log(V) = \sum_{i=1}^{f} l_i - \sum_{i=1}^{k-1} l'_i$.

The algorithm that we give is slightly different from the original given in [5]. We followed a suggestion of correction for this algorithm, given in [12], where:

1. In the original version, f is set to be $\min\{k + \beta, n\}$ and the intention was that the size of the set $\{k, \ldots, f\}$ was equal to β, so there is an element too much. The correction is to set $f = \min\{k + \beta - 1, n\}$;

2. The original version of the algorithm implicitly assumes that the l'_i's are initialized with the values stored in the l_i's at the beginning of the algorithm. This assumption is added explicitly in the recent version.

We remark as well that this algorithm may not work with bases of special structure, such as the bases of NTRU lattices, presented in the next chapter. A Java implementation of the BKZ simulation algorithm can be found in [12].

3.3 Public-Key Encryption Schemes

Like all already known classical public-key cryptosystems based on the hardness of certain number theory problems, such as RSA, elliptic curve cryptography (ECC), McEliece and many others, we can build many public-key encryption schemes using the presumed hardness of lattice problems. We will see three well-known proposals, the GGH/HNF, NTRU and LWE-based schemes, based on the hardness of the CVP, the SVP and LWE problem respectively.

Like lattice reduction algorithms, there is a duality between theory and practice in such encryption schemes. Some cryptosystems cannot be put in practice due to their inefficiency, yet interesting since they admit strong security proofs. This means that breaking such cryptosystems on the average case (i.e., when keys are chosen randomly) is at least as hard as solving the

underlying lattice problem in the worst case (i.e., solving the underlying lattice problem in any case). On the other hand, other cryptosystems notable by their efficiency in most cases do not enjoy of such security proofs.

We begin by describing the GGH cryptosystem with its HNF variant. This a very simple and interesting example of a lattice-based scheme, but proved to be unsafe.

After this, we introduce the NTRU scheme, which will be of our main interest. This is the most efficient cryptosystem known to date, but lacks a security proof. Besides being quantum-resistant, the NTRU cryptosystem is an alternative to the classical cryptosystems used nowadays, such as RSA. In this chapter we will introduce this public-key cryptosystem and in chapter 4 we will analyse its security, focusing obviously on lattice reduction attacks.

Finally we describe the LWE-based cryptosystem, proposed by Regev and based on the Learning With Errors problem. This is the most efficient cryptographic construction known to date relying on a security proof. The LWE-based cryptosystem is not as efficient as the NTRU cryptosystem, but it can be used in practice.

There are other cryptographic schemes with security proofs, such as the Ajtai-Dwork cryptosystem, which was the first one enjoying a security proof. However, this cryptosystem is not efficient: the public key size is $\tilde{O}(n^4)$ and the encryption blowup factor is $\tilde{O}(n^2)$. When the dimension of the underlying lattice is sufficiently high, the public key size can reach several gigabytes, making the cryptosystem useless in practice.

3.3.1 The GGH/HNF Public-Key Cryptosystem

In 1997, Goldreich, Goldwasser and Halevi published the GGH cryptosystem, which is the lattice analogue of the McEliece cryptosystem. Its idea is described in Scheme 1. The GGH

Scheme 1: The GGH/HNF Public-Key Cryptosystem

Parameters : A dimension n, a lattice $L \subseteq \mathbb{Z}^n$ of dimension n and $M \in \mathbb{N}$

Private Key: A *good* basis B of L formed by short and almost orthogonal vectors

Public Key : A *bad* basis H for L, such as the Hermite Normal Form (HNF) of B. If the HNF of B is taken, we refer this cryptosystem by GGH/HNF.

Encryption : The message space is the set $\{m = (m_1, \ldots, m_n) \in \mathbb{Z}^n : -M \leq m_1, \ldots, m_n \leq M\}$. Given an element m of the message space, the ciphertext is $c = Hm + e$, where e is a short noise vector.

Decryption : To decrypt the ciphertext c, one computes $B^{-1}c = B^{-1}(Hm + e) = B^{-1}(BUm + e) = Um + B^{-1}e$. The Babai rounding technique will be used to remove the term $B^{-1}e$ as long as it is small enough. Finally one computes $m = U^{-1}Um$.

cryptosystem, likewise NTRU and LWE-Based cryptosystems, is a probabilistic cryptosystem since a single message can be encrypted in many different ways depending on the chosen noise vector.

For a full-rank matrix $B \in \mathbb{Z}^{n \times n}$, we define the Hermite Normal Form of B to be the matrix $H \in \mathbb{Z}^{n \times n}$ satisfying the following conditions:

1. $H_{i,j} = 0$ for all $j > i$ (i.e., H is a lower triangular matrix);

2. $H_{i,i} \geq 0$ for all i (i.e., the pivot of each column of H is positive);

3. $H_{i,j} \geq 0$ and $H_{i,j} \leq H_{i,i}$ for all $j < i$.

The HNF of a matrix always exists and is unique. There are alternative definitions of the HNF of a matrix, such as defining it with upper triangular matrices, although the result is equal up to transposition. The HNF of a matrix can be efficiently computed using the NTL Library. As we can see in the description of the algorithm, it is useful to store the unimodular matrix U such that $H = BU$.

Essentially, the Hermite Normal Form is good to be used as a public basis. This is because if one can attack the HNF H of the private basis B, then all bases B' of the lattice can be attacked by computing theirs HNF H' which is H. In addition, the HNF of a *good* basis normally is not a *good* basis.

At Crypto '99 conference, Nguyen showed that the GGH cryptosystem is not secure, since every ciphertext can be used to uncover some information about the original message, and since decryption is easier than presupposed [24]. Still, GGH remains secure for sufficiently large values of n, though such values make GGH impractical.

3.3.2 The NTRU Public-Key Cryptosystem

The NTRU cryptosystem was firstly proposed in 1996 by Jeffrey Hoffstein, Jill Pipher and Joseph Silverman and later in 1998 published. The name NTRU is short for N^{th} degree truncated polynomial ring, but this acronym can also stand for *Number Theorists aRe Us* and *Number Theory Research Unit*.

Several descriptions of the NTRU cryptosystem can be found in the literature. Some describe it more generally, as it was introduced in [13] and [15], and others describe it in a particular and suitable case choice of parameters. In this description we will begin by presenting the general case and then step by step descending to those particular cases.

Notation and Parameters

Let N, p, q be positive integers. In order to fix notation and ease writing, we define

$$R_N = \mathbb{Z}[X]/\langle X^N - 1\rangle$$

and similarly

$$R_{N,p} = \mathbb{Z}[X]/\langle p, X^N - 1\rangle \quad \text{and} \quad R_{N,q} = \mathbb{Z}[X]/\langle q, X^N - 1\rangle.$$

From now on, the multiplication in R_N will be denoted by the symbol $\circledast$. We also define, for later purposes, the polynomial set

$$\mathcal{P}_p(c) = \left\{ \begin{array}{c} \text{polynomials of degree at most } c-1 \\ \text{with (mod } p\text{) coefficients} \end{array} \right\}$$

and the notation

$$[g]_p = \begin{array}{c} g \text{ with its coefficients reduced} \\ \text{modulo } p \text{ into the range }]-p/2, p/2] \end{array}$$

for all polynomial $g \in \mathbb{Z}[X]$.

Besides being polynomials, the elements of R_N can be expressed as vectors of coefficients, i.e., if $f \in R_N$,

$$f = \sum_{i=0}^{N-1} f_i X^i = (f_0, f_1, \ldots, f_{N-1}).$$

Since $\circledast$ is essentially a convolution product, it is easy to see that, if $f, g \in R_N$, then

$$f \circledast g = \sum_{k=0}^{N-1} h_k X^k,$$

where

$$\begin{aligned} h_k &= \sum_{i=0}^{k} f_i g_{k-i} + \sum_{i=k+1}^{N-1} f_i g_{N+k-i} \\ &= \sum_{i+j \equiv k \pmod N} f_i g_j. \end{aligned}$$

We define a generating function

$$G : \mathcal{P}_p(c) \to \mathcal{P}_p(c)$$

and a hash function

$$H : \mathcal{P}_p(c) \times \mathcal{P}_p(c) \to \mathcal{P}_p(c'),$$

required for building a *digital envelope*. In order to guarantee security, given an output value of these functions, we must ensure that the inputs are difficult to find. Moreover, to guarantee efficiency, G and H must be easy to compute.

The original version of NTRU given in [15] does not include the use of a digital envelope, i.e., both functions G and H would be identically zero. This obviously reduces the security of the cryptosystem. We may refer to this improved version of the original NTRU as *NTRU PKC digital envelope*.

Following the previous notation, the main parameters of the NTRU cryptosystem are N, p, q, where $\gcd(p, q) = 1$ and $q \gg p$, and an additional integer parameter $0 \leq k < N$. Note that p and q need not be prime. In section 4.1.1 we will explain why we need the assumption $\gcd(p, q) = 1$. In addition to these four integer parameters (N, p, q, k), the NTRU cryptosystem depends on four sets $\mathcal{L}_f, \mathcal{L}_g, \mathcal{L}_\phi \subseteq R_N$ and $\mathcal{L}_m \subseteq \mathcal{P}_p(N - k)$.

Key Creation

To create both public and private keys, we choose two polynomials $f \in \mathcal{L}_f$ and $g \in \mathcal{L}_g$. To avoid some confusion induced by the notation, we must read "$\mathcal{L}_f$ is the set where we pick polynomials playing the role of f". This is valid for the other three sets defined before. In addition, we must require that f has both inverses modulo p and modulo q. So if this is not the case, we must repeat the choice of f until both inverses exist.

When studying the efficiency of the cryptosystem, we must then ensure that the choice of such a polynomial f is quick, i.e., that finding such a polynomial with both inverses modulo p and q in $\mathcal{L}_f$ happens at the first attempt. The computation of these inverses can be done using the Euclidean algorithm, or using the *Almost Inverse Algorithm* of Schroeppel, Orman, O'Malley and Spatscheck [43]. We will study the invertibility in truncated polynomial rings in section 4.1.2.

These inverses are denoted by f_p^{-1} and f_q^{-1}, meaning that,

$$f_p^{-1} \circledast f \equiv 1 \pmod{p} \quad \text{and} \quad f_q^{-1} \circledast f \equiv 1 \pmod{q}.$$

Next, we compute

$$h \equiv p f_q^{-1} \circledast g \pmod{q}$$

which will be the public key. In [15], h is defined as $h \equiv f_q^{-1} \circledast g \pmod{q}$, but for efficiency reasons, due to ciphertext form, we decided to define h this way.

The private key is the polynomial f together with f_p^{-1}, since storing f_p^{-1} will be particularly useful for decryption. As we will see, after computing the public key h, one can discard the chosen polynomial g.

Encryption

Suppose that Bob has selected (N, p, q, k) as parameters and has settled h as his public key. If Alice wants to send a message $m \in \mathcal{L}_m \subseteq \mathcal{P}_p(N-k)$ to Bob, she chooses a polynomial $\phi \in \mathcal{L}_\phi$ randomly and computes the ciphertext

$$e \equiv \phi \circledast h + [m + H(m, [\phi \circledast h]_p) X^{N-k} + G([\phi \circledast h]_p)]_p \pmod{q},$$

which will be sent to Bob. When not using a digital envelope, the encrypted message is simply

$$e \equiv \phi \circledast h + m \pmod{q}.$$

As with the public key h, in [15] the ciphertext is defined as $e \equiv p\phi \circledast h + m \pmod{q}$.

Decryption

When Bob receives a ciphertext e from anyone, say Alice, he first computes

$$a = [f \circledast e]_q,$$

which is, by the previous notation, $f \circledast e$ reduced modulo q with coefficients in the interval $]-q/2, q/2]$. Then he computes

$$t = f_p^{-1} \circledast a \pmod{p}$$

and

$$b \equiv e - t \pmod{p} \quad \text{and} \quad c \equiv t - G(b) \pmod{p},$$

and after that he writes c in the form

$$c = c' + c''X^{N-k} \quad \text{with } \deg(c') < N - k \text{ and } \deg(c'') < k.$$

Lastly he compares c'' and $H(c', b)$: if these two values are equal, Bob has correctly decrypted the ciphertext and the original message is c'; if not, a decryption error has occurred.

Now we will explain why is this procedure able to decrypt a ciphertext. Developing the polynomial $f \circledast e \pmod{q}$, we get

$$\begin{aligned} f \circledast e &\equiv f \circledast \phi \circledast h + f \circledast [m + H(m, [\phi \circledast h]_p)X^{N-k} + G([\phi \circledast h]_p)]_p \pmod{q} \\ &\equiv f \circledast \phi \circledast pf_q^{-1} \circledast g + f \circledast [m + H(m, [\phi \circledast h]_p)X^{N-k} + G([\phi \circledast h]_p)]_p \pmod{q} \\ &\equiv p\phi \circledast g + f \circledast [m + H(m, [\phi \circledast h]_p)X^{N-k} + G([\phi \circledast h]_p)]_p \pmod{q} \end{aligned}$$

and therefore

$$a = \Big[p\phi \circledast g + f \circledast [m + H(m, [\phi \circledast h]_p)X^{N-k} + G([\phi \circledast h]_p)]_p\Big]_q.$$

Choosing a suitable set of parameters, the polynomial

$$p\phi \circledast g + f \circledast [m + H(m, [\phi \circledast h]_p)X^{N-k} + G([\phi \circledast h]_p)]_p$$

has coefficients in $]-q/2, q/2]$ for most ciphertexts e, and thus when Bob computes a he retrieves exactly

$$p\phi \circledast g + f \circledast [m + H(m, [\phi \circledast h]_p)X^{N-k} + G([\phi \circledast h]_p)]_p$$

in R_N. Reducing this polynomial modulo p he gets

$$f \circledast [m + H(m, [\phi \circledast h]_p)X^{N-k} + G([\phi \circledast h]_p)]_p$$

in R_N, and multiplying the result by f_p^{-1}, he gets

$$m + H(m, [\phi \circledast h]_p)X^{N-k} + G([\phi \circledast h]_p)$$

in $R_{N,p}$, which is the polynomial t. Hence

$$\begin{aligned} b &\equiv e - t \pmod{p} \\ &\equiv \phi \circledast h \pmod{p} \end{aligned}$$

and

$$\begin{aligned} c &\equiv t - G(b) \pmod{p} \\ &\equiv m + H(m, [\phi \circledast h]_p) X^{n-k} \pmod{p}. \end{aligned}$$

Therefore

$$c' \equiv m \pmod{p} \quad \text{and} \quad c'' \equiv H(m, [\phi \circledast h]_p) \pmod{p}.$$

Remark 5. 1. This decryption procedure has a chance to fail, usually when the message m is not centered, i.e., when the coefficients of m are far from the interval $]-q/2, q/2]$. If this is the case, Bob can try to choose a different interval $]-q/2 \pm x, q/2 \pm x]$ where the coefficients of a lie, a translation of the interval $]-q/2, q/2]$ with x small. If this attempt results in recovering the message, we say that a *wrap failure* occurred. If no value of x works, we say that a *gap failure* occurred. Therefore it is important to guarantee that the message space $\mathcal{L}_m$ contains only centered polynomials. However, more study about the parameters of NTRU is required in order to avoid decryption errors. We will see in section 4.1.3 what parameters ensure that decryption errors are very rare or nonexistent.

2. In [41] we can find an efficient mod q to mod p conversion algorithm, assisting the efficiency of the cryptosystem.

Parameter Selection

For efficiency reasons, we are interested in fast modulo q operations, so we set q being a power of 2. Since $\gcd(p, q) = 1$, we must then have p odd. In view of the previous remark, we set

$$\mathcal{L}_m = \left\{ m \in R_N : \begin{array}{c} m \text{ has coefficients between } -\frac{1}{2}(p-1) \text{ and } \frac{1}{2}(p-1) \\ \text{and has degree at most } N-k-1 \end{array} \right\}.$$

Again motivated by efficiency reasons, the polynomials in $\mathcal{L}_f, \mathcal{L}_g$ and $\mathcal{L}_\phi$ are wanted with small coefficients, so we set

$$\mathcal{L}_f = \mathcal{L}(d_f + 1, d_f), \quad \mathcal{L}_g = \mathcal{L}(d_g, d_g), \quad \text{and} \quad \mathcal{L}_\phi = \mathcal{L}(d_\phi, d_\phi),$$

where d_f, d_g, d_ϕ are positive integers and

$$\mathcal{L}(d_1, d_2) = \left\{ F \in R_N : \begin{array}{c} F \text{ has } d_1 \text{ coefficients equal to } 1, d_2 \text{ coefficients} \\ \text{equal to } -1, \text{ and the remaining coefficients all equal to } 0 \end{array} \right\}.$$

We set $\mathcal{L}_f = \mathcal{L}(d_f + 1, d_f)$ instead of $\mathcal{L}_f = \mathcal{L}(d_f, d_f)$, since the polynomial f must be invertible modulo p and q. If $f \in \mathcal{L}(d_f, d_f)$, then $f(1) = 0$, and then f cannot be invertible, as the next proposition says.

Proposition 5. *If $f \in R_N$ is invertible, then $f(1) \neq 0$. If $f \in R_{N,q}$ is invertible, then $f(1)$ is invertible in $\mathbb{Z}_q$, i.e., $f(1) \in \mathbb{Z}_q^*$.*

Proof. For the first part, write $f = f_0 + f_1 X + \cdots + f_{N-1} X^{N-1}$ and its inverse $f' = f'_0 + f'_1 X +$

$\cdots + f'_{N-1}X^{N-1}$. Hence

$$\begin{cases} f_0f'_0 & + & f_1f'_{N-1} & + & \cdots & + & f_{N-1}f'_1 & = & 1 \\ f_0f'_1 & + & f_1f'_0 & + & \cdots & + & f_{N-1}f'_2 & = & 0 \\ & & & & \vdots & & & & \\ f_0f'_{N-1} & + & f_1f'_{N-2} & + & \cdots & + & f_{N-1}f'_0 & = & 0. \end{cases}$$

Then,

$$f_0\left(\sum_{i=0}^{N-1} f'_i\right) + f_1\left(\sum_{i=0}^{N-1} f'_i\right) + \cdots + f_{N-1}\left(\sum_{i=0}^{N-1} f'_i\right) = 1,$$

so

$$\left(\sum_{i=0}^{N-1} f_i\right)\left(\sum_{i=0}^{N-1} f'_i\right) = 1,$$

hence

$$f(1)f'(1) = 1,$$

and therefore $f(1) \neq 0$.
For the second part the above calculations are valid modulo q, permitting to conclude that $f(1)f'(1) \equiv 1 \pmod{q}$. □

In the following table, we present the suggested parameters given in the two original NTRU papers [13] and [15]. If not using a digital envelope, one can ignore the value of k. These parameter choices lead to a residual probability of getting a decryption error, as we can see by experimentation.

Security	N	p	q	k	d_f	d_g	d_ϕ
Low	107	3	64	-	14	12	5
Moderate	167	3	128	49	60	20	18
High	263	3	128	52	49	24	16
Highest	503	3	256	107	215	72	55

Table 3.1: Security levels were adjusted to current attacks;
In the first parameter set, given in [15], k is not defined,
as the use of a digital envelope was not introduced yet.

Synthesis

Here is a compact of the previous sections:

Scheme 2: The NTRU Public-Key Cryptosystem

Parameters : $N, p, q, d_f, d_g, d_\phi$

Private Key: $f \in \mathcal{L}(d_f + 1, d_f)$ invertible in $R_{N,p}$ and $R_{N,q}$, $g \in \mathcal{L}(d_g, d_g)$

Public Key : $h \equiv p f_q^{-1} \circledast g \pmod q$

Encryption : $e = \phi \circledast h + m \pmod q$, where $\phi \in \mathcal{L}(d_\phi, d_\phi)$

Decryption : Compute $a \equiv f \circledast e \pmod q$ where the coefficients of a are in the interval $]-q/2, q/2]$.
Take $a \pmod p$ and compute $f_p^{-1} \circledast a \pmod p$ to retrieve the message m.

Scheme 3: The NTRU digital envelope Public-Key Cryptosystem

Parameters : $N, p, q, k, d_f, d_g, d_\phi$

Private Key: $f \in \mathcal{L}(d_f + 1, d_f)$ invertible in $R_{N,p}$ and $R_{N,q}$, $g \in \mathcal{L}(d_g, d_g)$

Public Key : $h \equiv p f_q^{-1} \circledast g \pmod q$

Encryption : $e = \phi \circledast h + [m + H(m, [\phi \circledast h]_p) X^{N-k} + G([\phi \circledast h]_p)]_p] \pmod q$,
where $\phi \in \mathcal{L}(d_\phi, d_\phi)$ and $m \in \mathcal{P}_p(N-k)$

Decryption : Compute $a \equiv f \circledast e \pmod q$ where the coefficients of a are in the interval $]-q/2, q/2]$.
Compute $t \equiv f_p^{-1} \circledast a \pmod p$.
Compute $b \equiv e - t \pmod p$ and $c \equiv t - G(b) \pmod p$.
Write c in the form $c = c' + c'' X^{N-k}$ with
$\deg(c') < N - k$ and $\deg(c'') < k$.
If $c'' \equiv H(c', b) \pmod p$, c' is the message m.
If not, a decryption error has occurred.

Current NTRUEncrypt

Naturally, the NTRU cryptosystem has undergone many changes over time, as we can for example witness in [18]. For completing this description of the NTRU cryptosystem, we present the actual parameter settings, key creation, encryption and decryption proceedings given in [18]. However, in the next chapter we only will focus on the first versions of the NTRU cryptosystem, described in the previous sections. Nevertheless, making this section seems appropriate, at least for showing the current state of art of NTRU.

Throughout this section we consider $p = 3$ and q a power of 2. We set the notation:

$$\begin{aligned} \mathcal{T}_N &= \{\text{trinary polynomials}\} \subseteq R_N \\ \mathcal{T}_N(d, e) &= \left\{ \begin{array}{c} \text{trinary polynomials with exactly} \\ d \text{ ones and } e \text{ minus ones} \end{array} \right\} \subseteq \mathcal{T}_N \\ \mathcal{P}_N(d_1, d_2, d_3) &= \left\{ A_1 \circledast A_2 + A_3 : A_i \in \mathcal{T}_N(d_i, d_i) \right\} \subseteq \mathcal{T}_N \end{aligned}$$

The private key is going to be $(f, g) = (1+pF, g)$, where $F \in \mathcal{P}_N(d_1, d_2, d_3)$ and $g \in \mathcal{T}_N(d_g+1, d_g)$ are randomly chosen. In order to build the public key, we must require that $f = 1 + pF$ is

invertible modulo q, which happens for a suitable choice of parameters. Choosing polynomials of the form $A_1 \circledast A_2 + A_3$ comes from efficiency reasons, since it allows fast multiplications.

Algorithm 6: NTRUEncrypt Key Generation

Input: NTRUEncrypt parameters $N, p, q, d_1, d_2, d_3, d_g$

Output: Private key (f, g) and Public key $(1, h)$

repeat
  $F \leftarrow \mathcal{P}_N(d_1, d_2, d_3)$
  $f = 1 + pF \in R_{N,q}$
until f is invertible in $R_{N,q}$
$g \leftarrow \mathcal{T}_N(d_g + 1, d_g)$
$h = f_q^{-1} \circledast g \in R_{N,q}$

To prevent combinatorial attacks, such as meet-in-the-middle attacks, we maximize the key space taking $d_g = \lfloor N/3 \rceil$ and $2d_1d_2 + d_3 \approx N/3$, since the expected number of non-zero coefficients in f is $4d_1d_2 + 2d_3$.

In order to encrypt a message $M \in \{0,1\}^l$, we consider three supporting functions: an invertible message formatting function FMT, a blinding polynomial generation function BGF and a mask generation function MGF:

$$\begin{aligned} \text{FMT}: &\quad \text{Message} \times \text{Random bits} \longrightarrow \mathcal{T}_N \\ \text{BGF}: &\quad \text{Message} \times \text{Random bits} \times \text{Public key} \longrightarrow \mathcal{P}_N(d_1, d_2, d_3) \\ \text{MGF}: &\quad R_{N,q} \longrightarrow \mathcal{T}_N \end{aligned}$$

The following algorithms instruct how to encrypt and decrypt. The details can be found in [8] and [14], and an implementation in [37].

Algorithm 7: NTRUEncrypt Encryption

Input: Public key h, message $M \in \{0,1\}^l$ and a parameter set with $p = 3$

Output: Ciphertext c

repeat
  $b \leftarrow \{0,1\}^t$
  $m' = \text{FMT}(M, b)$
  $r' = \text{BGF}(m', b, h)$
  $r = pr' \circledast h$
  $mask = \text{MGF}(r)$
  $m = m' + mask \pmod p$
until The number of $+1, -1$ and 0 in m are each $\geq d_m$
$c = r + m$

Algorithm 8: NTRUEncrypt Decryption

Input: Key pair $((f, g), (1, h))$, ciphertext $c \in R_{N,q}$ and a parameter set with $p = 3$

Output: *result*

$m = f \circledast c \pmod{p}$
$r = c - m$
$mask = \mathrm{MGF}(r)$
$m' = m - mask \pmod{p}$
$(M, b) = \mathrm{FMT}^{-1}(m')$
$r' = \mathrm{BGF}(m', b, h)$
if $pr' \circledast h = r$ and the number of $\{+1, -1, 0\}$ in m are each $\geq d_m$ **then**
  $result = M$
else
  $result = \perp$
end if

A new parameter d_m is used to assure that decryption has similar difficulty for different results of m. For more information about the parameter d_m, see [14].

Finally, in the following table also given in [14] we give the recommended parameter settings for this version of NTRUEncrypt and the security estimates for each set of parameters. The security estimate is essentially the number of steps needed for an attacker to break a private key or a message. In Section 4.2.2 we explicitly compute some examples. The decryption failure probability for each set of parameters is residual.

Parameter Sets and Security Estimates

Security estimate	N	q	d_1	d_2	d_3	d_g	d_m
116	401	2048	8	8	6	133	101
133	439	2048	9	8	5	146	112
193	593	2048	10	10	8	197	158
256	743	2048	11	11	15	247	204

3.3.3 The LWE-Based Cryptosystem

The LWE-Based cryptosystem was presented in 2005 by Regev together with a theoretical security proof, unlike both GGH and NTRU. As previously stated, this kind of cryptosystem is mostly impractical, but this is not the case. This is why this cryptosystem is inescapable when studying lattice-based cryptography. In Scheme 4 we describe the cryptosystem.

Scheme 4: The LWE-based public-key cryptosystem

Parameters : Integers n, m, l, t, r, q and a real $\alpha > 0$.
The parameter n is the main security parameter.

Private Key: A matrix $S \in \mathbb{Z}_q^{n \times l}$ chosen uniformly at random.

Public Key : A pair $(A, P = AS + E) \in \mathbb{Z}_q^{m \times n} \times \mathbb{Z}_q^{m \times l}$ where $A \in \mathbb{Z}_q^{m \times n}$ is chosen uniformly at random and $E \in \mathbb{Z}_q^{m \times l}$ is chosen with entries according to Ψ_α, which is a distribution on $\mathbb{Z}_q$ defined as

$$\Psi_\alpha = \lfloor X \rceil \pmod q,$$

where $X \sim \mathcal{N}\left(0, \left(\frac{\alpha q}{\sqrt{2\pi}}\right)^2\right)$.

Encryption : The ciphertext is $(u = A^T a, c = P^T a + f(v)) \in \mathbb{Z}_q^n \times \mathbb{Z}_q^l$, where $v \in \mathbb{Z}_t^l$ is the message to encrypt, (A, P) is the public key, $a \in \{-r, -r+1, \ldots, r\}^m$ is chosen uniformly at random and

$$\begin{aligned} f \colon \mathbb{Z}_t^l &\to \mathbb{Z}_q^l \\ (x_1, \ldots, x_l) &\mapsto \left(\lfloor \frac{q}{t} x_1 \rceil, \ldots, \lfloor \frac{q}{t} x_l \rceil\right). \end{aligned}$$

Decryption : Compute $f^{-1}(c - S^T u)$, where $(u, c) \in \mathbb{Z}_q^n \times \mathbb{Z}_q^l$ is the ciphertext, $S \in \mathbb{Z}_q^{n \times l}$ is the private key and

$$\begin{aligned} f^{-1} \colon \mathbb{Z}_q^l &\to \mathbb{Z}_t^l \\ (x_1, \ldots, x_l) &\mapsto \left(\lfloor \frac{t}{q} x_1 \rceil, \ldots, \lfloor \frac{t}{q} x_l \rceil\right). \end{aligned}$$

Notice that f^{-1} defined by this way is not necessarily the inverse of f, it is just a notation.

The *learning with errors problem* (LWE) can be described in the following way: given $n, m, q \in \mathbb{Z}$, a probability distribution χ on $\mathbb{Z}_q$ (usually a *rounded* normal distribution) and a pair $(A, v) \in \mathbb{Z}_q^{m \times n} \times \mathbb{Z}_q^m$, find with some non-negligible probability if v was chosen uniformly from $\mathbb{Z}_q^m$ or chosen as $v = As + e$, where $s \in \mathbb{Z}_q^n$ is uniformly chosen and $e \in \mathbb{Z}_q^m$ is chosen according to χ^m.

Using the same notation as in Section 3.1, the LWE problem can be reformulated in the following way: find if v was chosen uniformly from $\mathbb{Z}_q^m$ or chosen by the perturbation (according to χ) of an arbitrary lattice point in $L_q(A)$.

Currently, no polynomial time algorithm is known for solving the LWE problem, yet there are some algorithms for solving it in exponential time. The next thing to do is to perceive what parameters should we take in order to ensure the efficiency and security of the cryptosystem, not forgetting the probability of decryption errors.

Efficiency

Following the instructions of the LWE-based cryptosystem in Scheme 4, we can observe that this is a cryptosystem easy to implement. The following table summarizes the LWE operating characteristics in terms of its parameters (all logarithms are base 2):

Private key size	$nl \log q$ bits
Public key size	$m(n+l) \log q$ bits
Message size	$l \log t$ bits
Ciphertext size	$(n+l) \log q$ bits
Encryption blowup factor[1]	$\left(1 + \frac{n}{l}\right) \log q / \log t$
Operations for encryption per bit	$\tilde{O}\left(m\left(1 + \frac{n}{l}\right)\right)$
Operations for decryption per bit	$\tilde{O}(n)$

As we can see in the table, the LWE-based cryptosystem has a weakness: the length of the public-key.

Decryption errors

Like all probabilistic encryption schemes, it is important to study the probability of decryption errors depending on the chosen parameters. By the description of the LWE-based cryptosystem, we have

$$\begin{aligned} f^{-1}(c - S^T u) &= f^{-1}(P^T a + f(v) - S^T A^T a) \\ &= f^{-1}((AS + E)^T a + f(v) - S^T A^T a) \\ &= f^{-1}(E^T a + f(v)). \end{aligned}$$

Now we will estimate the probability of a single letter of $v \in \mathbb{Z}_t$ get wrongly decrypted. Without generality loss, we consider the first letter of v.

Let $x \in \mathbb{Z}_t$ be the first coordinate of $E^T a$. In order to get a decryption error in v, we must have $|x| > \frac{q}{2t}$. Now suppose that a is fixed. Since the entries of E are defined according to Ψ_α defined above, we have

$$x \sim \mathcal{N}\left(0, \left(\frac{\alpha q ||a||}{\sqrt{2\pi}}\right)^2\right),$$

since the sum of independent normal variables is a normal distribution with mean the sum of means and variance the sum of variances.

We now estimate the value of $||a|| = \sqrt{a_1^2 + \cdots + a_m^2}$. For that, we compute the expected value of each a_i^2. Hence

$$E[a_i^2] = \sum_{k=-r}^{r} \frac{1}{2r+1} k^2 = \frac{r(r+1)}{3}.$$

[1] Also known as message expansion or ciphertext expansion.

Therefore

$$||a|| \approx \sqrt{\frac{r(r+1)m}{3}}.$$

Finally, after estimating $||a||$, we can say that

$$x \sim \mathcal{N}\left(0, \left(\alpha q \sqrt{\frac{r(r+1)m}{6\pi}}\right)^2\right)$$

and

$$\begin{aligned}\text{error probability per letter} &= P\left(|x| > \frac{q}{2t}\right) \\ &= 2\left(1 - \Phi\left(\frac{\frac{q}{2t} - 0}{\alpha q \sqrt{\frac{r(r+1)m}{6\pi}}}\right)\right) \\ &= 2\left(1 - \Phi\left(\frac{1}{2t\alpha} \cdot \sqrt{\frac{6\pi}{r(r+1)m}}\right)\right),\end{aligned}$$

where Φ is the cumulative distribution function (CDF) of the standard normal distribution $\mathcal{N}(0,1)$.

Security

The security guarantee is divided in two pieces. Given the difficulty and the irrelevancy of this part for the purposes of this work, we will not explain in detail the security analysis of the LWE-based cryptosystem.

Since the LWE-based cryptosystem is based on the hardness of the LWE problem, we must choose parameters n, m, q, Ψ_α for which the LWE problem is believed to be hard. With this choice of parameters, the public keys (A, P) generated by the LWE-based cryptosystem are indistinguishable from arbitrary uniform pairs. This is essentially the first part. The second one consists to ensure that no statistical information about the encrypted message can be obtained if one encrypts with a random public key (A, P).

Since the encrypted message is $(A^T a, P^T a + f(v)) \in \mathbb{Z}_q^n \times \mathbb{Z}_q^l$, the second part consists to ensure that $(A^T a, P^T a)$ is uniformly distributed. For this to happen, in [2] it is suggested to choose parameters verifying

$$(2r+1)^m \gg q^{n+l}.$$

For the first part, we use the reformulation of the LWE problem using lattices, where $\chi = \lfloor \Psi_\alpha \rceil$. To see if v is close to the lattice $L_q(A)$, we can use the dual lattice of $L_q(A)$ and see if $\langle v, w \rangle$ is close to an integer for a short vector $w \in L_q(A)$. If $v = x + e$ with $x \in L$ and e some small perturbation,

$$\langle v, w \rangle = \langle x + e, w \rangle = \underbrace{\langle x, w \rangle}_{\in \mathbb{Z}} + \langle e, w \rangle.$$

By Cauchy-Schwarz inequality,

$$|\langle e, w \rangle| \leq \|e\| \|w\|,$$

so $\langle v, w \rangle$ is close to an integer, namely $\langle x, w \rangle$, if e is small, i.e., v is close to L. However, this method is not very effective when the perturbation vector is of norm much bigger than $1/||w||$. Given that the perturbation vector follows a normal distribution of standard deviation $\alpha q/\sqrt{2\pi}$, in the drive to ensure security we must have

$$\frac{\alpha q}{\sqrt{2\pi}} \gg \frac{1}{||w||}.$$

From Section 3.1, we can predict that the shortest vector of $L_q(A)^* = \frac{1}{q} L_q^{\perp}(A^T)$ is of length

$$||w|| \approx q^{\frac{n}{m}-1} \cdot \sqrt{\frac{m}{2\pi e}}$$

and then we must ensure that

$$\alpha \gg \frac{2\pi\sqrt{e}}{q^{n/m}\sqrt{m}}.$$

Choice of parameters

In order to guarantee security, the parameters of the cryptosystem must satisfy the two conditions:

$$(2r+1)^m \gg q^{n+l};$$
$$\alpha \gg \frac{2\pi\sqrt{e}}{q^{n/m}\sqrt{m}}.$$

If $l \ll n$, the encryption blowup factor given by $\left(1 + \frac{n}{l}\right) \log q / \log t$ will be very large, so we have to discard this option. Choosing $l = n$ gives an acceptable balance between the encryption blowup factor and the public key size, which was the primary weakness of the LWE-based cryptosystem.

In order to choose a good set of parameters, we first set n, m, q, thus set α satisfying the second condition above, and finally set r satisfying the first equation. As said before, we take $l = n$. After that, it only remains to choose t. We highlight that the choice of parameters should not only guarantee security, but also optimize the operating characteristics of LWE. Further optimizations of the LWE cryptosystem can be found in [2].

Chapter 4

Analysis of NTRU Cryptosystem

4.1 Security Analysis

RSA may be attacked by factoring, so, to be secure, sufficiently large integers must be used so as to make factoring infeasible. Similarly, the most effective attacks against NTRU use algorithms that find short vectors in a lattice, so a NTRU cryptosystem must use parameters sufficiently large so as to make infeasible to find short vectors.

We begin by describing some elementary attacks, then we give a detailed description of the lattice attacks. We emphasize that some of the presented attacks do not work anymore with the current implementation of NTRU, e.g. the current implementation of NTRU is resistant to adaptive chosen ciphertext attacks, and thus decryption failure attacks such as the presented below will not work anymore.

We will work with the original NTRU cryptosystem as it is presented in [15], thus forgetting the parameter k and the generating and hash functions G and H, all introduced before. Many other attacks exist, but we present the best known and important ones.

4.1.1 Implementation considerations

When we introduced the NTRU cryptosystem, we defined the parameters p and q as positive integers satisfying $\gcd(p, q) = 1$. Let us observe why this assumption is very important for the security of the cryptosystem. Suppose that $\gcd(p, q) = d$. Since

$$\begin{aligned} e &\equiv \phi \circledast h + m \pmod{q} \\ &\equiv p\phi \circledast f_q^{-1} \circledast g + m \pmod{q}, \end{aligned}$$

we have that

$$q \,|\, (e - p\phi \circledast f_q^{-1} \circledast g - m)$$

and thus

$$e - p\phi \circledast f_q^{-1} \circledast g - m = qk$$

for some integer k. Hence

$$\begin{aligned} e - m &= qk + p\phi \circledast f_q^{-1} \circledast g \\ &= d\left(\frac{q}{d}k + \frac{p}{d}\phi \circledast f_q^{-1} \circledast g\right), \end{aligned}$$

and thus $d|(e-m)$ and $e \equiv m \pmod{d}$. Therefore reducing e modulo d allows an attacker to gather valuable information about the message m. Thus, it is important to make d as small as possible. In the worst case, when $d = p$, i.e., when $d|p$, $e \equiv m \pmod{p}$, which means that an attacker gets exactly the message m, making the cryptosystem completely insecure.

In addition, it is highly advised to choose N prime, since if N is composite, an attacker can recover the private key by solving a lattice problem in dimension lower than $2N$. For example, if N is even, we factorize $X^N - 1$ as

$$X^N - 1 = (X^{N/2} - 1)(X^{N/2} + 1)$$

and consider the natural maps

$$p_1 : \frac{\mathbb{Z}[X]}{\langle X^N - 1\rangle} \longrightarrow \frac{\mathbb{Z}[X]}{\langle X^{N/2} - 1\rangle} \quad \text{and} \quad p_2 : \frac{\mathbb{Z}[X]}{\langle X^N - 1\rangle} \longrightarrow \frac{\mathbb{Z}[X]}{\langle X^{N/2} + 1\rangle}.$$

Since the public key h is in $\mathbb{Z}[X]/\langle X^N - 1\rangle$, we consider $p_1(h)$ and $p_2(h)$ and the associated lattices defined in Section 4.1.9, say L_1 and L_2. If we can find private keys f_1 and f_2 associated to L_1 and L_2 respectively, using the Chinese Remainder Theorem, we get a private key $f \in \mathbb{Z}[X]/\langle X^N - 1\rangle$. Hence, if N is even, an attacker can recover the private key by solving a lattice problem of dimension N, and if $N/2$ is again even, an attacker can recover the private key by solving a lattice problem of dimension $N/4$.

Of course this argument is extensible to the case when N is composite. We cannot factorize $X^N - 1$ as simple as the case when N is even, but we can use the identity

$$X^N - 1 = \prod_{d|N} \Phi_d(X),$$

where $\Phi_d(X)$ is the d-th cyclotomic polynomial.
For example,

$$\begin{aligned} X^{15} - 1 \;=\; & (X-1)(X^2+X+1)(X^4+X^3+X^2+X+1) \\ & (X^8 - X^7 + X^5 - X^4 + X^3 - X + 1). \end{aligned}$$

4.1.2 Invertibility in truncated polynomial rings

Let us denote by $R_{N,q}^*$ the group of invertible elements of $R_{N,q}$, i.e.,

$$R_{N,q}^* = \{f \in R_{N,q} : f \circledast g = 1 \text{ for some } g \in R_{N,q}\}.$$

Since the NTRU cryptosystem uses invertible polynomials modulo p and q, for efficiency reasons it is important to guarantee that most polynomials in $R_{N,p}$ and $R_{N,q}$ are invertible. Thus we

will study the ratio $\#R^*_{N,q}/\#R_{N,q}$ for an arbitrary positive integer q.

The first thing to notice is that we can reduce this problem to the case when q is a power of a prime. This is due to the Chinese Remainder Theorem: if $q = q_1 q_2$ with $\gcd(q_1, q_2) = 1$, then

$$R_{N,q} \cong R_{N,q_1} \times R_{N,q_2} \quad \text{and} \quad R^*_{N,q} \cong R^*_{N,q_1} \times R^*_{N,q_2}.$$

Theorem 7. *Let p be a prime number, q a power of p, and $N \geq 2$ an integer such that $\gcd(p, N) = 1$.*
Let $n \geq 1$ be the smallest positive integer such that $p^n \equiv 1 \pmod{N}$. For each integer $d|n$, let

$$\nu_d = \frac{1}{d} \sum_{d'|d} \mu\left(\frac{d}{d'}\right) \gcd(N, p^{d'} - 1).$$

Then,

$$\frac{\#R^*_{N,q}}{\#R_{N,q}} = \prod_{d|n} \left(1 - \frac{1}{p^d}\right)^{\nu_d}.$$

In particular, if N is prime, then $\nu_d = 0$ for all $1 < d < n$, and

$$\frac{\#R^*_{N,q}}{\#R_{N,q}} = \left(1 - \frac{1}{p}\right)\left(1 - \frac{1}{p^n}\right)^{(N-1)/n}.$$

This theorem and its proof can be found in [42]. Notice also that the function μ in the theorem is the classical Möbius function.

Now consider the evaluation map

$$\begin{aligned} \mathrm{ev}_1 \colon R_{N,q} &\to \mathbb{Z}_q \\ f(X) &\mapsto f(1). \end{aligned}$$

It is a ring homomorphism, so it induces a group homomorphism

$$R^*_{N,q} \to \mathbb{Z}^*_q.$$

We know that

$$\mathbb{Z}^*_q \cong \{a \in \mathbb{Z}_q : \gcd(a, q) = 1\},$$

so if $\gcd(f(1), q) \neq 1$, then $f(1)$ is not invertible in $\mathbb{Z}_q$. Using Proposition 5 of Chapter 3, we conclude that if f is invertible in $R_{N,q}$, then $\gcd(f(1), q) = 1$.

Since $f \in \mathcal{L}(d_f + 1, d_f)$, we have $f(1) = 1$, so we will study the ratio $\#R^*_{N,q}(1)/\#R_{N,q}(1)$, where

$$R_{N,q}(1) = \{f \in R_{N,q} : f(1) = 1\} \quad \text{and} \quad R^*_{N,q}(1) = \{f \in R^*_{N,q} : f(1) = 1\}.$$

Remarking that $\#R_{N,q}(1) = q^{-1}\#R_{N,q}$, $\#R^*_{N,q}(1) = \phi(q)^{-1}\#R^*_{N,q}$ (where ϕ is Euler's totient function) and that if $q = p^k$, then $\phi(q) = p^k - p^{k-1}$, we deduce that

$$\frac{\#R^*_{N,q}(1)}{\#R_{N,q}(1)} = \left(1 - \frac{1}{p}\right)^{-1} \frac{\#R^*_{N,q}}{\#R_{N,q}}.$$

Since in practice p is small ($p = 2$ or $p = 3$) and N is prime, using Theorem 7 we get

$$\frac{\#R^*_{N,q}(1)}{\#R_{N,q}(1)} = \left(1 - \frac{1}{p^n}\right)^{(N-1)/n} \approx 1 - \frac{N-1}{np^n}.$$

Therefore, fixing N prime and a small value for p, namely $p = 2$ or $p = 3$, in order to maximize the above ratio n must be large, i.e., the order of p in $\mathbb{Z}_N$ must be large. Since $p \in \mathbb{Z}^*_N$ whose order is $\phi(N) = N-1$, the order of p divides $N-1$. Thus it is suggested to choose N prime such that $N-1$ has not many divisors. This suggests the use of Sophie Germain primes. Taking a Sophie Germain prime M, we choose $N = 2M+1$ which is prime (by the definition of a Sophie Germain prime), and consequently $N-1$ has as divisors $\{1, 2, M, 2M\}$. Hence if $N \nmid p^2 - 1$, then the order of p in $\mathbb{Z}_N$ is M or $2M$.

If $N = 2M + 1 \geq 167$, where M is a Sophie Germain prime, and $p \in \{2, 3\}$, then it is clear that the order of p in $\mathbb{Z}_N$ is M or $2M$. Therefore,

$$\frac{\#R^*_{N,q}(1)}{\#R_{N,q}(1)} \approx 1 - \frac{N-1}{Mp^M} = \frac{2M+1-1}{Mp^M} = 1 - \frac{2}{p^M}.$$

Since usually $N \geq 167$, we have $M \geq 83$, so this ratio is extremely close to 1.

4.1.3 Wrap and gap failures

We define, for a polynomial $f \in R_N$,

$$|f|_\infty = \max_{0 \leq i \leq N-1}\{f_i\} - \min_{0 \leq i \leq N-1}\{f_i\}$$

and

$$|f|_2 = \left(\sum_{i=0}^{N-1}(f_i - \mu_f)^2\right)^{1/2}, \quad \text{where} \quad \mu_f = \frac{1}{N}\sum_{i=0}^{N-1} f_i.$$

The value $|f|_\infty$ is called the *width* or the *spread* of f. Since the standard deviation of the coefficients of f is

$$\begin{aligned}\sigma_f &= \left(\frac{1}{N}\sum_{i=0}^{N-1}(f_i - \mu_f)^2\right)^{\frac{1}{2}} \\ &= \frac{1}{\sqrt{N}}\left(\sum_{i=0}^{N-1}(f_i - \mu_f)^2\right)^{\frac{1}{2}},\end{aligned}$$

we conclude that $|f|_2/\sqrt{N}$ is the standard deviation of the coefficients of f.

Remark 6. We can easily check that $|\ |_2$ and $|\ |_\infty$ are both seminorms, but not norms.

Proposition 6. *Let $N \in \mathbb{N}$. Then,*

$$\underset{\epsilon>0}{\forall}\ \underset{\gamma_1,\gamma_2\in\mathbb{R}^+}{\exists} : \underset{\text{randomly chosen}}{\underset{f,g\in R_N}{\forall}} \text{Prob}(\gamma_1|f|_2|g|_2 \leq |f \circledast g|_\infty \leq \gamma_2|f|_2|g|_2) > 1-\epsilon.$$

Remark 7. In this proposition, given in [13] and [15], it is clear that $f, g \in R_N$ are taken randomly with coefficients in $[-r, r] \cap \mathbb{Z}$ with a large r, since a uniform probability distribution over the integers does not exist. This proposition is relevant since for somewhat large values of N and very small values of ϵ the ratio γ_2/γ_1 is not very high. If it were not so, the proposition the estimation of $|f \circledast g|_\infty$ would be weak.

With this new notation, if $f \in \mathcal{L}_f(d_f+1, d_f)$, $g \in \mathcal{L}_g(d_g, d_g)$ and $\phi \in \mathcal{L}_\phi(d_\phi, d_\phi)$, we have

$$|f|_2 = \sqrt{2d_f + 1 - 1/N}, \quad |g|_2 = \sqrt{2d_g}, \quad |\phi|_2 = \sqrt{2d_\phi}.$$

Following the decryption scheme, we must have

$$|f \circledast m + p\phi \circledast g|_\infty < q,$$

since otherwise the coefficients of $f \circledast m + p\phi \circledast g$ do not certainly lie in the interval $]-q/2, q/2]$. Taking

$$|f \circledast m|_\infty \leq q/4 \quad \text{and} \quad |p\phi \circledast g|_\infty \leq q/4,$$

the previous equation holds since $|\ |_\infty$ is a seminorm and because of that the triangular inequality holds. Using Proposition 6 (with ϵ very small), we must take

$$|f|_2|m|_2 \approx q/4\gamma_2 \quad \text{and} \quad |\phi|_2|g|_2 \approx q/4p\gamma_2.$$

For $N = 107, N = 167$ and $N = 503$, our experiments led to γ_2 values 0.89, 0.72 and 0.44 respectively. These values were obtained by programming Proposition 6 and they are different from the ones given in [13] and [15].

The following proposition describes a set of parameters that prevent decryption errors.

Proposition 7. *Consider the NTRU parameters $N, p, q, d_f, d_g, d_\phi$. If $q > (6d+1)p$, where $d = d_f = d_g = d_\phi$, then no decryption error can occur.*

Proof. Consider the polynomial $p\phi \circledast g + f \circledast m \in R_N$. For a decryption error never to occur, we must ensure that the coefficients of this polynomial lie in the interval $]-q/2, q/2]$. By the definition of the convolution product $\circledast$, the largest possible coefficient of $\phi \circledast g$ is $2d$. By the same reason, since the coefficients of m are between $-\frac{1}{2}p$ and $\frac{1}{2}p$, the largest possible coefficient of $f \circledast m$ is $(2d+1) \cdot \frac{1}{2}p$. Then the largest possible coefficient of $p\phi \circledast g + f \circledast m$ is

$$p \cdot 2d + (2d+1) \cdot \frac{1}{2}p = p \cdot \frac{1}{2}(6d+1).$$

Thus, as $q > (6d+1)p$, we get that every coefficient of $p\phi \circledast g + f \circledast m$ is $< q/2$. This argument is also valid for the least possible coefficient of $p\phi \circledast g + f \circledast m$. □

Remark 8. Since in the above proposition we did a rough estimate for the coefficients of $\phi \circledast g$ and $f \circledast m$, we can actually choose q slightly smaller than $(6d+1)p$, and this allows to reduce key sizes. However, to choose q significantly smaller than $(6d+1)p$ it is important to study the probability of a decryption failure. This probability must be null or residual, say 2^{-80} or less.

Notice that, if settled a NTRU scheme, Bob works as a decryption oracle, telling to everyone who sends a message to him if the message was decrypted correctly or was considered invalid. Then, it is possible to make an attack directed to Bob's private key through chosen ciphertexts sent to him, expecting a decryption failure to occur. To produce such an attack, Eve tries different pairs (m, ϕ) until her message gets rejected. Suppose that the pair (m, ϕ) produces a decryption error. Hence at least one coefficient of $p\phi \circledast g + f \circledast m$, say

$$c_i = \sum_{j=0}^{N-1} m_j f_{i-j} + p \sum_{j=0}^{N-1} \phi_j g_{i-j}$$

is not in the interval $]-q/2, q/2]$. It is important to remark that Eve does not know i. Now suppose that $p = 3$ and choose $j \in \{0, \ldots, N-1\}$.

- Supposing that $m_j = \pm 1$ and Eve changes it to 0, creating a new message m', if the pair (m', ϕ) produces a valid decryption then $f_{i-j} = \pm 1$.
  This is because if $f_{i-j} = 0$, then the new c_i is equal to the old c_i, and thus the new pair (m', ϕ) produces an invalid decryption, which is a contradiction.

- If (m, ϕ) produces a valid decryption with $m_j = 0$ but not with $m_j = \pm 1$, by the same argument as in the previous item we have $f_{i-j} = \pm 1$.

Therefore this attack can reveal important information about Bob's private key, specially when $p = 2$. It is thus a good practice to make the chance of a decryption error residual or null.

We can study the chance of getting wrap and gap failures by testing a large number of polynomials $a = p\phi \circledast g + f \circledast m \pmod q$. Running such experimentation, we can have an idea of the probability of a wrap failure. However, gap failures are so rare that even testing a large number of polynomials they may not show. If it happens, one can use the following formula, which is self-explanatory, to estimate the probability of a gap failure:

$$\text{Prob}\left(|a|_\infty \geq q\right) = \sum_j \text{Prob}\left(\max_{0 \leq i \leq N-1} \{a_i\} \geq j\right) \cdot \text{Prob}\left(\min_{0 \leq i \leq N-1} \{a_i\} = j - q\right).$$

In [46] we can find that the probability of having a wrap or a gap failure is

(N, p, q)	Probability of a wrap failure	Probability of a gap failure
$(107, 3, 64)$	$\approx 2^{-14}$	$\approx 2^{-29}$
$(167, 3, 128)$	$\approx 2^{-14}$	$\approx 2^{-29}$
$(263, 3, 128)$	$\approx 2^{-19}$	$\approx 2^{-41}$
$(503, 3, 256)$	$\approx 2^{-14}$	$\approx 2^{-28}$

where m has approximately the same number of zeros, ones and minus ones, and the calculation of the probability of having a gap failure was computed with the previous formula.

In [47], centering methods and estimate decryption failure probabilities are given, testing the theory for a typical parameter set $(N, p, q) = (251, 2, 239)$, with $f = 1+pF$, $F, g, \phi \in \mathbb{Z}_2[X]/\langle X^N-1\rangle$ with exactly 72 ones and $N-72$ zeros, $m \in \mathbb{Z}_2[X]/\langle X^N-1\rangle$ with exactly 125 ones and $N-125$ zeros, together with a centering method given in the same report. The results between theory and practice show good agreement.

4.1.4 Alternate Private Keys

It is possible that another polynomial than f in R_N can work as a private key, hence decrypt encrypted messages as f can do.

Let f' be a rotation of f, i.e., $f' = X^k \circledast f$ for some integer k. We know that $pg \equiv f \circledast h \pmod q$, thus $X^k \circledast pg \equiv X^k f \circledast h \pmod q$. Therefore, f' can decrypt the same messages as f:

Let $g' = X^k \circledast g$.
Since $pg \equiv f \circledast h \pmod q$, $pg' \equiv f' \circledast h \pmod q$.
Assume that we want to decrypt the ciphertext e.
We will compute

$$\begin{aligned} a' &\equiv f' \circledast e \pmod q \\ &\equiv f' \circledast (\phi \circledast h + m) \pmod q \\ &\equiv X^k \circledast f \circledast (\phi \circledast p f_q^{-1} \circledast g + m) \pmod q \\ &\equiv X^k \circledast (f \circledast \phi \circledast p f_q^{-1} \circledast g + f \circledast m) \pmod q \\ &\equiv X^k \circledast (\underbrace{p\phi \circledast g + f \circledast m}_{a}) \pmod q \\ &\equiv p\phi \circledast g' + f' \circledast m \pmod q. \end{aligned}$$

Since $a' = X^k \circledast a$, if all the coefficients of a lie in the interval $]-q/2, q/2]$, then so do the coefficients of a'.
Consider now f_p^{-1}, and its rotation $X^{-k} \circledast f_p^{-1}$.
We compute

$$\begin{aligned} (X^{-k} \circledast f_p^{-1}) \circledast a' &\equiv X^{-k} \circledast f_p^{-1} \circledast p\phi \circledast g' + X^{-k} \circledast f_p^{-1} \circledast f' \circledast m \pmod q \\ &\equiv m \pmod p. \end{aligned}$$

Therefore, we recovered the message m whose ciphertext is e.

In general, any pair (f', g') such that $pg' \equiv f' \circledast h \pmod q$ has an inverse modulo p can be used to decrypt messages if the coefficients of $a' \equiv p\phi \circledast g' + f' \circledast m \pmod q$ are in the interval $]-q/2, q/2]$. These pairs (f', g') are called *alternate keys*.

4.1.5 Brute force attacks

As the name implies, this attack consists on testing all possible private keys f, g or all possible random polynomials ϕ.

On the one hand, since $g \in \mathcal{L}(d_g, d_g)$ has small coefficients, an attacker can try all possible polynomials $f \in \mathcal{L}(d_f+1, d_f)$ and see if $p^{-1}f \circledast h \pmod q$ has small coefficients. On the other hand, since $f \in \mathcal{L}(d_f+1, d_f)$ has small coefficients, an attacker can also try all possible polynomials $g \in \mathcal{L}(d_g, d_g)$ and see if $pg \circledast h^{-1} \pmod q$ has small coefficients. Notice that h may not have an inverse modulo q.

Besides, since $m \in \mathcal{L}_m$ has small values, an attacker can try all possible polynomials $\phi \in \mathcal{L}(d_\phi, d_\phi)$ and see if $e - \phi \circledast h$ has small coefficients.

By Table 3.1 of Chapter 3, $\#\mathcal{L}_g < \#\mathcal{L}_f$, so the key security is determined by the parameter set $\mathcal{L}_g = \mathcal{L}(d_g, d_g)$. Besides, the message security is determined by $\mathcal{L}_\phi = \mathcal{L}(d_\phi, d_\phi)$. The next presented attacks (meet-in-the-middle attacks) will cut the search spaces $\mathcal{L}_g$ and $\mathcal{L}_\phi$ by the square root. Therefore, taking only into account brute force and meet-in-the-middle attacks, we have:

$$\begin{matrix}\text{Key}\\ \text{Security}\end{matrix} = \sqrt{\#\mathcal{L}_g} = \sqrt{\binom{N}{d_g}\binom{N-d_g}{d_g}} = \frac{1}{d_g!}\sqrt{\frac{N!}{(N-2d_g)!}}$$

$$\begin{matrix}\text{Message}\\ \text{Security}\end{matrix} = \sqrt{\#\mathcal{L}_\phi} = \sqrt{\binom{N}{d_\phi}\binom{N-d_\phi}{d_\phi}} = \frac{1}{d_\phi!}\sqrt{\frac{N!}{(N-2d_\phi)!}}$$

4.1.6 Meet-in-the-middle attacks

The following attack to the private key f was suggested by Andrew Odlyzko, although originally the attack was directed to ϕ. It can be found in [20] and is presented on random binary keys, although this attack can be adapted for other *-ary* values. Therefore the private key f has no longer $d_f + 1$ ones and d_f -1's, but $2d_f + 1$ ones. In [20], it is suggested to suppose that f has an even number of ones. However, with this assumption, f will not have an inverse modulo q, which is supposed to be a power of 2. Therefore, for explaining this attack we will suppose that f, g are binary, f has an odd number d of ones and $N-d$ zeros, and q is a power of 2.

Begin on literally splitting f in half, say $f = f_1 + f_2$, where f_1 is of length $\lfloor N/2 \rfloor$ and f_2 is of length $\lceil N/2 \rceil$. For example, if $f(X) = X^6+X^5+X^4+X^3+1$, then we get $f_1(X) = X^6+X^5+X^4$ and $f_2(X) = X^3 + 1$.

Next, if f_1 has $\lfloor d/2 \rfloor$ ones and f_2 has $\lceil d/2 \rceil$ ones, we can proceed to next step. If not, we need to rotate f in order to satisfy this condition. In the example above, this condition is not satisfied, but the rotations $f'(X) = X^2 f(X) = X^6 + X^5 + X^2 + X + 1$ and $f''(X) = X^5 f(X) = X^5 + X^4 + X^3 + X^2 + X$ of f satisfy it. We can always guarantee that at least one rotation of f has this property.

The idea of this attack is to search for f in the form $f_1 + f_2$, where f_1 and f_2 satisfy the above conditions. For this section we will use the description of h given in [15], $h \equiv f_q^{-1} \circledast g \pmod q$, instead of $h \equiv p f_q^{-1} \circledast g \pmod q$, since encryption is not necessary here and since the following can be easily adapted. Since $f \circledast h \equiv g \pmod q$, we have $(f_1 + f_2) \circledast h \equiv g \pmod q$, and then $f_1 \circledast h \equiv g - f_2 \circledast h \pmod q$, so $(f_1 \circledast h)_i = \{0, 1\} - (f_2 \circledast h)_i \pmod q$ for all $i = 0, \ldots, N-1$. Notice that the coefficients of $f_1 \circledast h$ and $f_2 \circledast h$ will differ by either 0 or 1 modulo q. If we

can find f' such that $f' \circledast h \pmod q$ is binary, then we have found at least one rotation of f or another key that is able to decrypt messages.

Once we have split f in the above conditions, we do:

1. **Enumerate f_1**
   There are $\binom{\lfloor N/2 \rfloor}{\lfloor d/2 \rfloor}$ polynomials of length $\lfloor N/2 \rfloor$ with exactly $\lfloor d/2 \rfloor$ ones and the rest of its coefficients equal to zero.
   For a fixed k, we put each f_1 into a *bin* $[b_1 \cdots b_k]$ in the following way:
   (a) Take the first k coefficients of $f_1 \circledast h \pmod q$ (in order);
   (b) Write them in base 2 (the length of the binary representation must be $\log_2(q)$, so in some cases we must add zeros to the left);
   (c) Consider the most significant bit of each of these base 2 representations (in order) and store them in the bin.

   where k is an integer chosen in such a way that $2^k > \binom{N/2}{d/2}$ (the number of bins superior than the number of possible f_1's).
   For example, suppose that $N = 4$, $q = 8$, $d = 3$, $k = 4$ and $f_1 \circledast h \pmod q = \mathbf{7} + \mathbf{2}X + \mathbf{3}X^2 + \mathbf{5}X^3 + 6X^4$.
   Writing $7 = (\mathbf{1}11)_2$, $2 = (\mathbf{0}10)_2$, $3 = (\mathbf{0}11)_2$, $5 = (\mathbf{1}01)_2$, the polynomial f_1 will be stored in the bin [1001].

2. **Enumerate f_2**
   There are $\binom{\lceil N/2 \rceil}{\lceil d/2 \rceil}$ polynomials of length $\lceil N/2 \rceil$ with exactly $\lceil d/2 \rceil$ ones and the rest of its coefficients equal to zero.
   For each f_2 we need to check if $-f_2 \circledast h$ corresponds to a bin occupied by a f_1.
   That means that the leading bits of the first k coefficients of $-f_2 \circledast h \pmod q$ are equal to the ones of $f_1 \circledast h \pmod q$.
   We want to find f_1 and f_2 such that $(f_1 \circledast h)_i \equiv \{0, 1\} - (f_2 \circledast h)_i \pmod q$. As a result, we want to check the bins that correspond to $-f_2 \circledast h$ and also the bins that correspond to adding ones to any of the coefficients of $-f_2 \circledast h$.
   For example, consider $-f_2 \circledast h \pmod q = \mathbf{7} + \mathbf{2}X + \mathbf{3}X^2 + \mathbf{5}X^3 + 6X^4$ as in the previous step. The polynomial f_2 will be stored in the bin [1001].
   If we add one to seven, we get eight (which is zero modulo 8) and the leading bit changes from one to zero. If we add one to two, we get three, and the leading bit remains equal to zero. If we add one to three, we get four, and the leading bit changes from zero to one. Finally, if we add one to five, we get six and the leading bit remains equal to one.
   Therefore, we have to check the bin [1001] and the bins [0001], [1011], [0011] as well.

3. **Search for matches**
   Once we have found that f_2 match with an occupied bin, we compute $f = f_1 + f_2$ and check if the coefficients of $f \circledast h \pmod q$ are binary.
   If it is binary, then the algorithm terminates and we have found a private key.
   If it is not binary, then we proceed to the next f_2.
   If the bin is occupied by more than one f_1, then we need to check f_2 for each f_1 in the bin.

At this point, we give an example to make this attack clear: consider $(N, d, q) = (6, 5, 8)$, $f(X) = X^5 + X^3 + X^2 + X + 1$ and $g(X) = X^3 + X + 1$. Then we define $f_1(X) = X^5 + X^3$ and $f_2(X) = X^2 + 1$. The public key is $h(X) = f_q^{-1} \circledast g \pmod q = 6X^5 + 7X^4 + 6X^3 + 6X^2 + 7X + 7$.

There are exactly $\binom{3}{2} = 3$ polynomials of length 3 with exactly 2 ones and the rest of the coefficients equal to zero. That is, $f_1 \in \{X^5 + X^4, X^5 + X^3, X^4 + X^3\}$. We will denote these three polynomials by $f_1^{(1)}, f_1^{(2)}$ and $f_1^{(3)}$.
We choose $k = 4$ and compute $f_1^{(j)} \circledast h \pmod q$ for each $j \in \{1, 2, 3\}$. Then, following the previous examples in the algorithm, the three $f_1^{(1)}, f_1^{(2)}$ and $f_1^{(3)}$ will be stored in the same bin [1111].
There are exactly $\binom{3}{3} = 1$ polynomial f_2 to list, namely $f_2^{(1)} = X^2 + X + 1$, which will be stored in the bin [1111].
For step 3, we compute

$$\begin{aligned} f^{(1)} &= f_1^{(1)} + f_2^{(1)} = X^5 + X^4 + X^2 + X + 1, \\ f^{(2)} &= f_1^{(2)} + f_2^{(1)} = X^5 + X^3 + X^2 + X + 1, \\ f^{(3)} &= f_1^{(3)} + f_2^{(1)} = X^4 + X^3 + X^2 + X + 1. \end{aligned}$$

Then, we compute

$$\begin{aligned} f^{(1)} \circledast h &= 1 + X^2 + X^5, \\ f^{(2)} \circledast h &= 1 + X + X^3, \\ f^{(3)} \circledast h &= X + X^2 + X^4. \end{aligned}$$

All of these polynomials are binary, then all of them can be used as private keys. Notice that $f^{(2)}$ is equal to f. In fact, these three polynomials are rotations of each other.

We now describe the running time of this algorithm. The first important remark to do is that the expected running time decreases as we increase k, however, increasing k costs increasing memory use. Following some remarks in [20] we can improve this algorithm, and the final estimate of the running time and storage space required for this method is $\frac{1}{\sqrt{N}}\binom{N/2}{d/2}$.

This attack applies also to the case when the private key f is of the form $f = f_1 \circledast f_2$, $f = f_1 \circledast f_2 + f_3$ or $f = 1 + pF$, where F is binary or takes one of the product forms described by the two previous cases. The idea is also given in [20].

4.1.7 Multiple transmission attacks

Suppose that Alice sends to Bob the same message r times, i.e., that Alice sends to Bob the r ciphertexts

$$e_i \equiv \phi_i \circledast h + m \pmod q, \quad i \in \{1, \ldots, r\}.$$

This is a bad practice, since an attacker can recover large parts of the message m. Suppose that Eve knows that Alice has sent the same message m multiple times, say r times. She then computes

$$\begin{aligned} c_i &\equiv (e_i - e_1) \circledast h^{-1} \pmod q \\ &\equiv e_i \circledast h^{-1} - e_1 \circledast h^{-1} \pmod q \\ &\equiv \phi_i + m \circledast h^{-1} - \phi_1 - m \circledast h^{-1} \pmod q \end{aligned}$$

$$\equiv \phi_i - \phi_1 \pmod{q}.$$

for every $i \in \{1, \ldots, r\}$.

Remark 9. Notice again that h may not have an inverse modulo q. However, there exists h' such that $a \circledast h \circledast h' \equiv a \pmod{q}$ for all polynomials a satisfying $a(1) = 0$. This polynomial h' is called a *pseudo-inverse* of h, and we can use it in the above calculation.

Since the coefficients of $\phi_i - \phi_1$ are between -2 and 2, when Eve computes c_i she recovers exactly $\phi_i - \phi_1$, rather than modulo q. For $j \in \{0, \ldots, N-1\}$, let α, β and γ be the j^{th} coefficient of ϕ_i, ϕ_1 and $\phi_i - \phi_1$, respectively. Then, the possible values for $\gamma = \alpha - \beta$ are:

		β		
		-1	0	1
α	-1	0	-1	-2
	0	1	0	-1
	1	2	1	0

Hence Eve can deduce the following:

- If $\gamma = \pm 2$, then $\beta = \mp - 1$;
- If $\gamma = \pm 1$, then $\beta \in \{0, \mp 1\}$.

Therefore,

- If $\gamma = \pm 2$, then it is possible to recover more or less $2/9$ of the coefficients of ϕ_1;
- If $\gamma = \pm 1$, then it is possible to obtain valuable information about $4/9$ of the coefficients of ϕ_1.

It is then clear that sending the same message, even if a few times, can reveal to an attacker large parts of ϕ_1. In this fashion Eve can recover large parts of m.

However, for this attack to work, Eve has to know if Alice has sent the same message m multiple times. This is not an issue, since

$$(e_i - e_j) \circledast h^{-1} \equiv (\phi_i - \phi_j) + (m_i - m_j) \circledast h^{-1} \pmod{q}$$

has small coefficients if and only if m_i and m_j are the same (because $\phi_i - \phi_j$ has small coefficients and h^{-1} in general has more dispersed coefficients).

We can find in [17] two ways to prevent multiple transmission attacks.

4.1.8 Semantic security

Recall the ciphertext and public key forms

$$e \equiv \phi \circledast h + m \pmod{q},$$

$$h \equiv p f_q^{-1} \circledast g \pmod{q}.$$

Since $g(1) = d_g - d_g = 0$, we have

$$h(1) \equiv 0 \pmod{q}.$$

Therefore,

$$\begin{aligned} e(1) &\equiv m(1) \pmod{q} \\ &\equiv m_0 + \cdots + m_{N-1} \pmod{q}. \end{aligned}$$

Because of this, an attacker can acquire relevant information about the message m by just computing $e(1)$.

4.1.9 Lattice-based attacks

Lattice-based attacks are currently the most effective ones against the NTRU cryptosystem. Lattices are good candidates to make an attack, since the private key f and messages m have small coefficients. Then, using lattice reduction algorithms such as LLL and BKZ, we may find private keys (or an alternate keys) or messages. The two first presented attacks (also called *hybrid attacks*) are by far the most effective ones, hence they are the main focus of study.

Lattice attack on an NTRU private key

The *standard NTRU lattice* L^{NT} is defined as the lattice generated by the columns of the matrix

$$\left(\begin{array}{cccc|cccc} \alpha & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 \\ 0 & \alpha & \cdots & 0 & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \alpha & 0 & 0 & \cdots & 0 \\ \hline h'_0 & h'_{N-1} & \cdots & h'_1 & q & 0 & \cdots & 0 \\ h'_1 & h'_0 & \cdots & h'_2 & 0 & q & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ h'_{N-1} & h'_{N-2} & \cdots & h'_0 & 0 & 0 & \cdots & q \end{array}\right)_{2N \times 2N},$$

where $h'_i \equiv p^{-1} h_i \pmod{q}$. This matrix is composed by four $N \times N$ blocks, so the lattice L^{NT} is of dimension $2N$. In the matrix, $\alpha \in \mathbb{R}$ is a short parameter to be chosen in order to maximize the efficiency of lattice reduction algorithms.

Since $h \equiv pf_q^{-1} \circledast g \pmod{q}$, the vector $\tau = (\alpha f, g)$ is in the lattice L^{NT}:

$$\left(\begin{array}{cccc|cccc} \alpha & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 \\ 0 & \alpha & \cdots & 0 & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \alpha & 0 & 0 & \cdots & 0 \\ \hline h'_0 & h'_{N-1} & \cdots & h'_1 & q & 0 & \cdots & 0 \\ h'_1 & h'_0 & \cdots & h'_2 & 0 & q & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ h'_{N-1} & h'_{N-2} & \cdots & h'_0 & 0 & 0 & \cdots & q \end{array}\right)\left(\begin{array}{c} f_0 \\ f_1 \\ \vdots \\ f_{N-1} \\ \hline 0 \\ 0 \\ \vdots \\ 0 \end{array}\right) \equiv \left(\begin{array}{c} \alpha f_0 \\ \alpha f_1 \\ \vdots \\ \alpha f_{N-1} \\ \hline g_0 \\ g_1 \\ \vdots \\ g_{N-1} \end{array}\right) \pmod{q}.$$

Therefore, our target vector will be $\tau = (\alpha f, g)$, which is a short vector in the lattice L^{NT}. Using the Gaussian Heuristic given in Chapter 3, the shortest vector in L^{NT} has length

$$s \approx \sqrt{\frac{N\alpha q}{\pi e}},$$

since $\det(L^{\mathrm{NT}}) = \alpha^N q^N$

Hence, lattice reduction algorithms have more chance to find the short vector τ if we maximize the ratio $s/\|\tau\|$. Since

$$\|\tau\|^2 = \alpha^2(2d_f + 1) + 2d_g \approx \alpha^2|f|_2^2 + |g|_2^2 = |\tau|_2^2,$$

we have

$$\begin{aligned} s^2/\|\tau\|^2 &\approx s^2/|\tau|_2^2 \\ &\approx \frac{\frac{N\alpha q}{\pi e}}{\alpha^2|f|_2^2 + |g|_2^2 + \frac{1}{2N}\alpha^2} \\ &\approx \frac{Nq}{\pi e} \cdot \frac{\alpha}{\alpha^2|f|_2^2 + |g|_2^2} \end{aligned}$$

Thus, in order to maximize the ratio $s/\|\tau\|$, we maximize

$$f(\alpha) = \frac{\alpha}{\alpha^2|f|_2^2 + |g|_2^2}.$$

Therefore, after computing the derivative of $f(\alpha)$, an attacker must choose

$$\alpha = |g|_2/|f|_2 = \sqrt{\frac{2d_g}{2d_f + 1 - \frac{1}{N}}}.$$

Defining c_h as the ratio $\|\tau\|/s$ with $\alpha = |g|_2/|f|_2$, we get

$$c_h \approx \sqrt{\frac{2\pi e|f|_2|g|_2}{Nq}}.$$

Notice that c_h is the inverse of the ratio that we have approximated before. Thus, this value tells to an attacker how easy the private key can be found: if the ratio c_h is small, it will be easy to find the private key. If c_h is close to 1, the private key lattice will be difficult to find. Since an attacker can compute c_h, Bob has to ensure that c_h is close to 1.

Lattice attack on an NTRU message

Very similar to a lattice attack on an NTRU private key, we consider the lattice of dimension $2N+1$ generated by the columns of the matrix

$$\left(\begin{array}{cccc|cccc|c} \alpha & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & 0 \\ 0 & \alpha & \cdots & 0 & 0 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & \alpha & 0 & 0 & \cdots & 0 & 0 \\ \hline h_0 & h_{N-1} & \cdots & h_1 & q & 0 & \cdots & 0 & e_0 \\ h_1 & h_0 & \cdots & h_2 & 0 & q & \cdots & 0 & e_1 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ h_{N-1} & h_{N-2} & \cdots & h_0 & 0 & 0 & \cdots & q & e_{N-1} \\ \hline 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & 1 \end{array}\right).$$

In contrast with a lattice attack on an NTRU private key, this lattice has dimension increased by 1, so this attack is in general less effective.

The target vector will be $(\alpha\phi, -m, -1)$, as

$$\left(\begin{array}{cccc|cccc|c} \alpha & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & 0 \\ 0 & \alpha & \cdots & 0 & 0 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & \alpha & 0 & 0 & \cdots & 0 & 0 \\ \hline h_0 & h_{N-1} & \cdots & h_1 & q & 0 & \cdots & 0 & e_0 \\ h_1 & h_0 & \cdots & h_2 & 0 & q & \cdots & 0 & e_1 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ h_{N-1} & h_{N-2} & \cdots & h_0 & 0 & 0 & \cdots & q & e_{N-1} \\ \hline 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & 1 \end{array}\right)\left(\begin{array}{c} \phi_0 \\ \phi_1 \\ \vdots \\ \phi_{N-1} \\ \hline 0 \\ 0 \\ \vdots \\ 0 \\ \hline -1 \end{array}\right) = \left(\begin{array}{c} \alpha\phi_0 \\ \alpha\phi_1 \\ \vdots \\ \alpha\phi_{N-1} \\ \hline -m_0 \\ -m_1 \\ \vdots \\ -m_{N-1} \\ \hline -1 \end{array}\right).$$

Doing the same calculations as before, we can conclude that an attacker must choose $\alpha = |\phi|_2/|m|_2$, and the constant

$$c_m \approx \sqrt{\frac{2\pi e|m|_2|\phi|_2}{Nq}}$$

tells to him how easy is to find a message: the smaller c_m gets, the easier it will be to find the message. If c_m is close to 1, the message will be difficult to find. Thus Alice must choose her messages in such a way that c_m is close to 1.

In conclusion, to make lattice attacks on the private key as difficult as lattice attacks on messa-

ges, Alice must choose messages (and random polynomials) satisfying $c_m \approx c_h$, i.e., $|f|_2|g|_2 \approx |m|_2|\phi|_2$. This potentially constitutes a limitation to Alice, but when $p = 3$, $|m|_2 \approx \sqrt{2N/3}$ and $|\phi|_2 = \sqrt{2d_\phi}$. Then, it is suggested that Bob choose

$$|f|_2|g|_2 \approx \sqrt{\frac{4}{3}Nd_\phi}.$$

and by this way Alice is free to choose the messages she wants.

The formulas

$$|f|_2|m|_2 \approx q/4\gamma_2 \quad \text{and} \quad |\phi|_2|g|_2 \approx q/4p\gamma_2$$

given in Section 4.1.3 can also contribute to the parameter choices.

Lattice attack on a spurious key

This attack, suggested by Coppersmith and Shamir in [6], consists on finding polynomials f' that can act as the private key.

If an attacker knows a short vector in the lattice L^{NT}, say $\tau' = (\alpha f', g')$, then if τ' is short enough, it can be used for decryption purposes, i.e., the polynomial $p\phi \circledast g' + f' \circledast m$ has coefficients lying in the interval $]-q/2, q/2]$ with high probability.

However, experimental evidence suggests that finding spurious keys f' takes not much less time than finding the private key f.

Zero-Run lattices

Since the polynomials f and g have a large number of zeros, and consequently $\tau = (\alpha f, g)$ has a large number of zeros, the following attack, suggested by Alexander May, proposes to take advantage of this fact.

The attack can be described by the following way:

1. Choose a very large number θ and a positive integer $r \leq 2N - 2d_f - 2d_g - 1$;
2. Pick r rows of the matrix of the lattice L^{NT} and multiply them by θ.
   The idea is to force lattice reduction algorithms to find short vectors with r-coordinates equal to 0, since otherwise those coordinates will have very large values, making the vectors found by the lattice reduction algorithms not to be small;
3. Use a lattice reduction algorithm to find short vectors in the new lattice, hoping to find the private key f (if one chooses the right r rows of the matrix of the lattice) or an alternate key.

This attack originally suggested the choice of r consecutive rows, but the key creator can make sure that the private key f has not a large number of consecutive zeros (anywhere in f, since a rotation of f can work as an alternate key). It is therefore advised to take r random rows instead of r consecutive ones.

Zero-forced lattices

The idea of using zero-forced lattices emerged from zero-run lattice attacks, and are way more efficient, since the dimension of zero-forced lattices is less than the lattice L^{NT}.

The attack can be described by the following way:

1. Choose a positive integer $r \leq N - 2d_g$ and a set of indexes $J = \{j_1, \ldots, j_r\}$, where $0 \leq j_1 < \cdots < j_r < N$.
   The idea is to force zeros in g, but one can on the other hand to force zeros in f (though in general g has more zeros than f) or in both f and g at the same time.
   Hence we will search short vectors with J-coordinates equal to 0.

2. Since we have the system of congruences

$$\begin{cases} f_0h'_0 + f_1h'_{N-1} + \cdots + f_{N-1}h'_1 \equiv g_0 \pmod q \\ f_0h'_1 + f_1h'_0 + \cdots + f_{N-1}h'_2 \equiv g_1 \pmod q \\ \vdots \quad\quad \vdots \quad\quad \ddots \quad\quad \vdots \quad\quad \vdots \\ f_0h'_{N-1} + f_1h'_{N-2} + \cdots + f_{N-1}h'_0 \equiv g_{N-1} \pmod q \end{cases}$$

   and $g_{j_1} = \cdots = g_{j_r} = 0$, we have the following r linear congruences:

$$\begin{cases} f_0h'_{j_1} + f_1h'_{j_1-1} + \cdots + f_{N-1}h'_{j_1-N+1} \equiv 0 \pmod q \\ f_0h'_{j_2} + f_1h'_{j_2-1} + \cdots + f_{N-1}h'_{j_2-N+1} \equiv 0 \pmod q \\ \vdots \quad\quad \vdots \quad\quad \ddots \quad\quad \vdots \quad\quad \vdots \\ f_0h'_{j_r} + f_1h'_{j_r-1} + \cdots + f_{N-1}h'_{j_r-N+1} \equiv 0 \pmod q. \end{cases}$$

   Solving this system of congruences for $f_{N-r}, \ldots, f_{N-1}$ and substituting in the first system, we get the system of congruences

$$\left\{ a_{j,0}f_0 + a_{j,1}f_1 + \cdots + a_{j,N-r-1}f_{N-r-1} \equiv g_j \pmod q \right.$$

   for $j \notin J$.
   We have solve the system of congruences for $f_{N-r}, \ldots, f_{N-1}$, but we can solve for any other r variables $f_{i_1}, \ldots, f_{i_r}$, where $0 \leq i_1 \leq \cdots \leq i_r \leq N-1$.

3. Now we define the Zero-Forced Lattice

$$L_J^{\mathrm{ZF}} = \left(\begin{array}{cccc|cccc} \alpha & 0 & \cdots & 0 & 0 & 0 & 0 & 0 \\ 0 & \alpha & \cdots & 0 & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & 0 & 0 & 0 & 0 \\ 0 & 0 & \cdots & \alpha & 0 & 0 & 0 & 0 \\ \hline a_{j'_0,0} & a_{j'_0,1} & \cdots & a_{j'_0,N-r-1} & q & 0 & \cdots & 0 \\ a_{j'_1,0} & a_{j'_1,1} & \cdots & a_{j'_1,N-r-1} & 0 & q & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{j'_{N-r-1},0} & a_{j'_{N-r-1},1} & \cdots & a_{j'_{N-r-1},N-r-1} & 0 & 0 & \cdots & q \end{array}\right),$$

where $0 \leq j'_0 < \cdots < j'_{N-r-1} < N$ and $\{j'_0, \ldots, j'_{N-r-1}\} = \{0, \ldots, N-1\} \setminus J$.
This is a lattice of dimension $2(N-r)$, and notice that the parameter α is in general different from the one of L^{NT}, i.e., the parameter α must be recomputed.
The target vector is $v_J = (\alpha f_0, \ldots, \alpha f_{N-r-1}, g_{j'_0}, \ldots, g_{j'_{N-r-1}})$, which will reveal the unknown values $g_{j'_0}, \ldots, g_{j'_{N-r-1}}$.

Remark 10. If we choose r too large, then J will be too large and the chance of choosing a good set J will be reduced. More information about zero-forced lattices can be found in [44]. We can find that this attack, using the lattice L^{NT} together with a zero-forced lattice, is the best attack against the NTRU cryptosystem. Although in [44] it is remarked that this kind of attack does not affect too much NTRU with $N = 167$ (see Table 3.1), currently this NTRU scheme can be broken in a fair amount of time.

4.2 Practical implementations of NTRU

4.2.1 Theoretical operating specifications

Considering the parameters (N, p, q, k), the NTRU cryptosystem has the following operating characteristics:

Private Key Size	$2N \log_2 p$ bits
Public Key Size	$N \log_2 q$ bits
Message Size	$(N-k) \log_2 p$ bits
Ciphertext Size	$N \log_2 q$ bits
Encryption Blowup Factor	$\frac{N}{N-k} \log_p q$
Encryption Speed	$O(N^2)$ operations
Decryption Speed	$O(N^2)$ operations

In section 4.3, we make some remarks about the encryption blowup factor (the message expansion) and about both encryption and decryption speeds.

Concerning encryption and decryption speeds, there are way more efficient ways to compute a product of polynomials $f \circledast g$ in R_N. The usual and naive method require N^2 multiplications, but since f and g have a large number of zeros, this is a fast multiplication. As suggested in [45], splitting f and g into two parts can reduce this multiplication to $\frac{3}{4}N^2$ operations. If this technique is repeated recursively, a smaller constant can be achieved. If N is sufficiently large, it may be faster to use Fast Fourier Transforms. This multiplication takes then $O(N \log N)$ operations.

4.2.2 Specific parameter choices

Let us focus on the parameters suggested in [13] and [15], presented in section 3.3.2. With the help of the above table, we analyze the meet-in-the-middle security levels of each parameter set and compute the lattice values c_h, c_m and s introduced before. We will write s in function of q in view of the next section.

We will exclude the parameter set with $N = 107$, as actually it is easily broken with lattice-based attacks.

	Moderate Security	High Security	Highest Security
(N, p, q, k)	$(167, 3, 128, 49)$	$(263, 3, 128, 52)$	$(503, 3, 256, 107)$
(d_f, d_g, d_ϕ)	$(60, 20, 18)$	$(49, 24, 16)$	$(215, 72, 55)$
Private Key Size	530 bits	834 bits	1595 bits
Public Key Size	1169 bits	1841 bits	4024 bits
Message Size	187 bits	335 bits	628 bits
Ciphertext Size	1169 bits	1841 bits	4024 bits
Encryption Blowup Factor	6.25	5.50	6.41
Key Security	$2^{82.9}$	$2^{110.6}$	2^{285}
Message Security	$2^{77.5}$	$2^{82.1}$	2^{170}
(c_h, c_m)	$(0.236, 0.225)$	$(0.187, 0.195)$	$(0.182, 0.160)$
s	$0.296q$	$0.409q$	$0.365q$

4.2.3 Lattice attacks - Experimental evidence and remarks

To study the NTRU lattices, we use the BKZ algorithm implemented in the NTL Library [15] in order to find a very short vector in the lattice L^{NT}. In practice, for the first values of β, BKZ does not find something better than a q-vector, i.e., a vector with one coordinate $= q$ and the rest equal to zero. Since the lattice L^{NT} already contains q-vectors, BKZ with small blocksizes is useless.

The blocksize value needed to pass this threshold, which depends obviously on N and c_h, appears to grow linearly with N, but we have to keep in mind that increasing β will significantly increase the running time of BKZ. The BKZ simulation algorithm is helpful to find an optimal trade-off between running time and quality of reduction.

When this threshold is passed, BKZ finds very short vectors in L^{NT}. If the current blocksize β does not allow to find a vector with length $\approx s$, then the following $\beta + 1$ has a good chance of finding such a vector. In general,

$$s \approx \sqrt{\frac{N\alpha q}{\pi e}}$$

is lesser than q, but not much lesser, hence after the threshold is passed there is a good chance of BKZ to find a vector in L^{NT} with length $\approx s$.

With this, the BKZ algorithm can either output a very short vector in L^{NT} or a q-vector which is useless. This allows to conclude that spurious keys are rare to find, since either we got a q-vector or a very short vector that can be the private key. This is why spurious keys do not constitute a threat to NTRU.

As observed in [15], [19] and [30], letting $t(N)$ denote the necessary time to find the private key of an NTRU setting with parameter N, the *graph* of $\log(t(N))$ seems to be, in average, something like a line with positive slope and some small positive concavity. This suggests that $t(N)$ grows at least exponentially with N, possibly with $N \log N$.

Remark 11. It is implied that $t(N)$ is not a function, since there are multiple NTRU settings

with the same main parameter N. For a fixed N, we consider some NTRU parameters and compute $t(N)$, which may vary a lot. Doing this for all (or some) reasonable N, we obtain a *graph* that consists in a *cloud* of points, which has the shape of a line with positive slope and small positive concavity.

In order to extrapolate $t(N)$ for larger values of N, we use linear regression to find constants a, b such that $\log(t(N)) \approx aN + b$. Such constants can be found in [15], [19] and [30].

Therefore, it remains an open problem how to break NTRU keys with large N in feasible time.

4.3 Additional topics

4.3.1 Improving message expansion

The parameter settings given in Sections 3.3.2 and 4.2.2 lead to somewhat large message expansions (namely between 5-to-1 and 7-to-1). It is thus important to find techniques that allow to suppress this weakness of NTRU.

There is a simple way to reduce the encryption blowup factor $log_p(q)$ of NTRU. Instead of considering a message with coefficients between $-(p-1)/2$ and $(p-1)/2$, Alice chooses a message with coefficients in $]-q/2, q/2]$ (in particular, trinary messages can be chosen). She randomly chooses $\phi_1 \in \mathcal{L}_\phi$ and a polynomial ϕ_2 with coefficients between $-(p-1)/2$ and $(p-1)/2$ and computes the quantities

$$\begin{aligned} e_2 &\equiv \phi_2 \circledast h + m \pmod q \\ e_1 &\equiv \phi_1 \circledast h + \phi_2 \pmod q \end{aligned}$$

and send (e_1, e_2) to Bob. The intention is to make ϕ_2 playing the message role. Hence, Bob can recover ϕ_2, and therefore can also recover

$$m \equiv e_2 - \phi_2 \circledast h \pmod q.$$

With this, the encryption blowup factor is reduced to 2 (although the ciphertext size is doubled).

4.3.2 Comparison with other cryptosystems

In this section we will compare some operating characteristics of well-known classical encryption schemes with the lattice-based ones. More specifically, we will compare NTRU, RSA, McEliece, GGH and ECC.

McEliece cryptosystem was developed by Robert McEliece in 1978, and is based on coding theory. In what concerns encryption and decryption, this cryptosystem is very fast when compared with ECC and RSA, but both private and public keys have large sizes, making it difficult to be used in practice. Despite of this, this cryptosystem is important since it is quantum-resistant.

The ciphertext form of the McEliece cryptosystem is of the form $E = AM + Y$ where A is the public key, M is the message and Y is a random element. The NTRU encryption can also be

described matricially:

$$\begin{pmatrix} 0 \\ e \end{pmatrix} \equiv L^{\mathrm{NT}} \begin{pmatrix} \phi \\ \psi \end{pmatrix} + \begin{pmatrix} -\phi \\ m \end{pmatrix} \pmod{q},$$

with the parameter α in L^{NT} equal to 1. Therefore, we observe that encrypting with NTRU is very similar to encrypting with McEliece, with random element and message exchanged.

The decryption procedure with NTRU is very different from all other decryption methods. Both McEliece and GGH decrypt by eliminating some small random perturbation, but NTRU has ciphertext form $e \equiv \phi \circledast h + m \pmod{q}$, so decryption eliminates the contribution $\phi \circledast h \pmod{q}$ which is in general not small. In addition, RSA and NTRU show almost nothing in common.

One advantage of NTRU is the length of both public and private keys, which produces a clear distinction with GGH (and McEliece of course). If a GGH lattice has dimension d, then key sizes are $O(d^2)$. If a NTRU lattice has dimension d, then the private key size is $O(d)$ and the public key size is $O(d \log d)$. Hence NTRU produces significantly smaller private and public keys than GGH, the size of whose public and private keys make the cryptosystem impractical when d is sufficiently large.

To be clear, we present the following table, suggested in [14], where we can find the operating features of the compared cryptosystems above, together with ECC. In the table, N is the main security parameter or the message length.

	NTRU	RSA	McEliece	GGH	ECC
Encryption Speed [1]	N^2	N^2	N^2	N^2	N^3
Decryption Speed [2]	N^2	N^3	N^2	N^2	N^3
Public Key	N	N	N^2	N^2	N
Private Key	N	N	N^2	N^2	N
Message Expansion [3]	varies	1-1	2-1	1-1	2-1

As verified in [22], regarding key sizes, ECC is always better than NTRU and RSA for security levels greater than 80 bits. Still about key sizes, under a security level of 160 bits RSA is better than NTRU, but for greatest security levels NTRU has a better performance than RSA. We point that the minimum security level considered as *strong security* is 80 bits. Keys that have a security level of 128 bits are recommended because they offer more security.

Regarding key generation, encryption and decryption (for the same security levels), the performance of NTRU is better than RSA. NTRU is also much faster than ECC with all levels of security in terms of key generation, encryption and decryption. However, McEliece is faster than NTRU regarding encryption and decryption times.

[1] RSA encryption is $O(N^3)$ unless small encryption exponents are used.

[2] NTRU encryption and decryption are asymptotically $O(N \log N)$ using FFTs.

[3] For NTRU, see sections 4.2.1 and 4.3.1. For McEliece, it is usually between 1.25 and 1.5.

Chapter 5

Prospect of Future Works

In this chapter we present a simple idea for a quantum algorithm that we have developed, but due to time issues we could not analyze it in detail. Besides, we briefly mention some topics of work to be done in the future that could contribute both for lattice reduction attacks and other attacks for NTRU cryptosystem.

The quantum algorithm that we present can be done with various polynomial time lattice reduction algorithms, but to ease writing we only point the LLL algorithm, whose implementation is found almost everywhere (NTL, Sage and Mathematica, for example).

The major benefit of this quantum algorithm is the superposition of all possible vectors written with a certain amount of memory, then run the LLL algorithm for a lattice of basis B together with this superposition of vectors. The summary of our work can be translated in the next three figures:

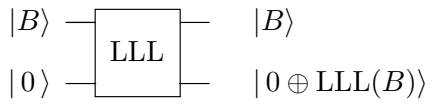

Fig. 5.1: LLL algorihtm as a quantum circuit. Note that in first set of wires we prepare the input basis B. On the second set we store the answer, which is *xored* with the initial value of the second set of wires, all initialized to 0.

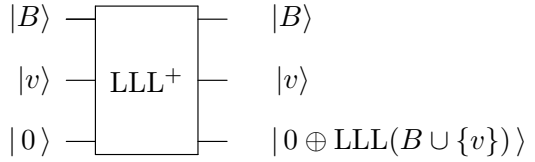

Fig. 5.2: Extension of the LLL algorithm as a quantum circuit where we consider an extra set of wires to code an ancilla vector v. We join this vector to the basis B and then apply LLL algorithm to the resulting matrix.

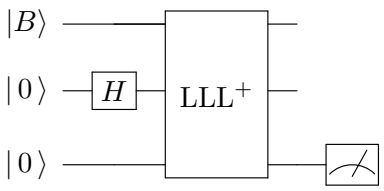

Fig. 5.3: Finally, we use the extended LLL with uniform superposition of all possible vectors. The uniform superposition is made with a Hadamard gate, creating all possible vectors written with a certain finite amount of memory. Then we run the extended LLL algorithm and measure the answer.

The idea is that, given a basis B of a lattice L, try all possible linear combinations of the vectors that generate L. This is thus a quantum brute force search.

We first define an interval where the coefficients of the linear combination will be, such as $\{-3, -2, -1, 0, 1, 2, 3, 4\}$. The cardinality of this interval must be a power of 2. It is very important to make this interval somewhat centered, as for example, if these coefficients were all positive then we would never get a shorter vector than the ones that we already know.

In general, if the lattice L is of dimension n and if we choose the length of the above interval to be equal to 2^m, then we get exactly 2^{mn} vectors. With this, we do the superposition of all these vectors, apply the LLL^+ algorithm to this superposition, and finally measure the answer.

The next step is to study the probability of this algorithm giving a useful answer. Of course, this algorithm can be done in a classical computer, although the running time would be exponential in the length of the chosen interval (and also in the dimension of the lattice), making it impractical, as we have to apply LLL to all vectors individually. Choosing a polynomial time reduction algorithm is also important. However, choosing for example the BKZ algorithm, we can abort the search after a certain amount of time and measure the answer.

Another perspective is to try to reduce lattice problems to other ones that can be efficiently dealt by a quantum computer, such as problems involving period finding, that can be dealt with QFTs, as it is done in [28]. On the other hand, to study in more depth classical lattice attacks in random lattices and lattices of a special form. The aim is to find new and more effective lattice attacks, for both random lattices and lattices of a special form, such as the NTRU lattice L^{NT}.

A subject to certainly explore is the *family* of NTRU, that is, some variants of the NTRU cryptosystem. NTRU is based on the ring $R_N = \mathbb{Z}[X]/\langle X^N - 1\rangle$, and the following table given in [49] summarizes some variants and the associated rings.

ETRU is based on the ring $\mathbb{Z}[\omega][X]/\langle X^N - 1\rangle$ where $\mathbb{Z}[\omega]$ is the ring of Eisenstein integers. The Eisenstein integers are the complex numbers of the form $z = a + b\omega$, where $a, b \in \mathbb{Z}$ and $\omega = e^{2\pi i/3}$. The Eisenstein integers form a triangular lattice in the complex plane.

QTRU is based on quaternion algebra and the associated ring is $\mathcal{O}_Q[X]/\langle X^N - 1\rangle$, where $\mathcal{O}_Q = \mathbb{Z} \cdot 1 \oplus \mathbb{Z} \cdot i \oplus \mathbb{Z} \cdot j \oplus \mathbb{Z} \cdot k$.

NTRU Prime [3] introduce rings of the form $\mathbb{Z}[X]/\langle X^p - X - 1\rangle$, where p is a prime number. It was proposed by Daniel Bernstein, Tanja Lange *et al.* in May 2016.

Variant of NTRU	**Ring**
ETRU	$\mathbb{Z}[\omega][X]/\langle X^N - 1\rangle$, $\omega = e^{2\pi i/3}$
QTRU	$\mathcal{O}_Q[X]/\langle X^N - 1\rangle$
CTRU	$\mathbb{F}_2[T][X]/\langle X^N - 1\rangle$
pNE	$\mathbb{Z}[X]/\langle X^{2^N} + 1\rangle$
NTWO	$\mathbb{Z}[X,Y]/\langle X^N - 1, Y^N - 1\rangle$
Non-commutative NTRU	$\mathbb{Z}[D_n][X]/\langle X^N - 1\rangle$, D_n dihedral group
Matrix NTRU	$\mathbb{Z}^{m\times m}$
MaTRU	$\big(\mathbb{Z}[X]/\langle X^N - 1\rangle\big)^{m\times m}$
NNRU	$\mathbb{Z}^{m\times m}[X]/\langle X^N - I_m\rangle$
NTRU Prime	$\mathbb{Z}[X]/\langle X^p - X - 1\rangle$, p prime

References

1. Ajtai, M., Kumar, R., & Sivakumar, D. (2001). A sieve algorithm for the shortest lattice vector problem. *Proceedings of the Thirty-third Annual ACM Symposium on Theory of Computing - STOC '01*.

2. Bernstein, D. J., Buchmann, J., & Dahmén, E. (2009). *Post-quantum cryptography*. Berlin: Springer.

3. Bernstein, D. J., Chuengsatiansup, C., Lange, T., & van Vredendaal, C. (2016). NTRU prime. *IACR Cryptology ePrint Archive*, 2016/461.

4. Chen, L., Jordan, S., Liu, Y., Moody, D., Peralta, R., Perlner, R., & Smith-Tone, D. (2016). Report on Post-Quantum Cryptography. *NISTIR 8105 DRAFT*. National Institute of Standards and Technology, U.S. Department of Commerce.

5. Chen, Y., & Nguyen, P. Q. (2011). BKZ 2.0: Better Lattice Security Estimates. *Lecture Notes in Computer Science Advances in Cryptology – ASIACRYPT 2011*, 1-20.

6. Coppersmith, D., & Shamir, A. (1997). Lattice Attacks on NTRU. *Advances in Cryptology — EUROCRYPT '97 Lecture Notes in Computer Science*, 52-61.

7. Dagdelen, O., & Schneider, M. (2010). Parallel Enumeration of Shortest Lattice Vectors. *Euro-Par 2010 - Parallel Processing Lecture Notes in Computer Science*, 211-222.

8. Efficient Embedded Security Standards (2015). EESS #1: Implementation aspects of NTRUEncrypt. *Consortium for Efficient Embedded Security*, Version 3.1.

9. Gama, N., & Nguyen, P. Q. (2008). Finding short lattice vectors within Mordell's inequality. *Proceedings of the Fourtieth Annual ACM Symposium on Theory of Computing - STOC 08*.

10. Gama, N., & Nguyen, P. Q. (2008). Predicting Lattice Reduction. *Advances in Cryptology – EUROCRYPT 2008 Lecture Notes in Computer Science*, 31-51.

11. Gama, N., Nguyen, P. Q., & Regev, O. (2010). Lattice Enumeration Using Extreme Pruning. *Advances in Cryptology – EUROCRYPT 2010 Lecture Notes in Computer Science*, 257-278.

12. Hamann, T. (2013). *The BKZ simulation algorithm* (Bachelor thesis). Technische Universität Darmstadt.

13. Hoffstein, J., Lieman, D., Pipher, J., & Silverman, J. H. (1999). NTRU: A public key cryptosystem. Available at `http://grouper.ieee.org/groups/1363/lattPK/submissions/ntru.pdf`.

14. Hoffstein, J., Pipher, J., Schanck, J. M., Silverman, J. H., Whyte, W., & Zhang, Z. (2015). Choosing parameters for NTRUEncrypt. *IACR Cryptology ePrint Archive*, 2015/708.

15. Hoffstein, J., Pipher, J., & Silverman, J. H. (1998). NTRU: A ring-based public key cryptosystem. *Lecture Notes in Computer Science Algorithmic Number Theory*, 267-288.

16. Hoffstein, J., Pipher, J., & Silverman, J. H. (2008). *An introduction to mathematical cryptography.* New York: Springer.

17. Hoffstein, J., & Silverman, J. H. (1998). *Implementation notes for NTRU PKCS multiple transmissions* (Tech. Rep. No. 006). NTRU Cryptosystems.

18. Hoffstein, J., & Silverman, J. H. (2000). Optimizations for NTRU. *Proceedings of the International Conference Organized by the Stefan Banach International Mathematical Center Warsaw, Poland, September 11-15, 2000 Public-Key Cryptography and Computational Number Theory.*

19. Hoffstein, J., Silverman, J. H., & Whyte, W. (2003). *Estimated breaking times for NTRU lattices* (Tech. Rep. No. 012). NTRU Cryptosystems.

20. Howgrave-Graham, N., Silverman, J. H., & Whyte, W. (2003). *A meet-in-the-middle attack on an NTRU private key* (Tech. Rep. No. 004, Version 2). NTRU Cryptosystems.

21. Lenstra, A. K., Lenstra, H. W., & Lovász, L. (1982). Factoring polynomials with rational coefficients. *Mathematische Annalen Math. Ann., 261*(4), 515-534.

22. Lindner, R., Rueckert, M., Baumann, P. & Nobach, L. (n.d.). TU Darmstadt Lattice Challenge. Retrieved August 15, 2016, from `https://www.latticechallenge.org/`.

23. Nguyen, H. B. (2014). *An overview of the NTRU cryptographic system* (Master of Arts thesis in Mathematics). San Diego State University.

24. Nguyen, P. Q. (1999). Cryptanalysis of the Goldreich-Goldwasser-Halevi Cryptosystem from Crypto '97. *Advances in Cryptology — CRYPTO '99 Lecture Notes in Computer Science*, 288-304.

25. Nguyen, P. Q. (2011). Lattice Reduction Algorithms: Theory and Practice. *Advances in Cryptology – EUROCRYPT 2011 Lecture Notes in Computer Science*, 2-6.

26. Nielsen, M. A., & Chuang, I. L. (2010). *Quantum computation and quantum information.* Cambridge: Cambridge University Press.

27. NTRU Cryptosystems, Inc. (2005). Peer review and independent scrutiny of the NTRUEncrypt public key cryptosystem. Available at `https://assets.securityinnovation.com/static/downloads/NTRU/resources/Ntru_Peer_Review.pdf`.

28. Regev, O. (2004). Quantum Computation and Lattice Problems. SIAM J. Comput. *SIAM Journal on Computing, 33*(3), 738-760.

29. Regev, O., & Kaplan, E. (2004). *LLL Algorithm* (Lecture notes of the course *Lattices in Computer Science*, Lecture 2). Tel Aviv University.

30. Sakshaug, H. (2007). *Security analysis of the NTRUEncrypt public key encryption scheme* (Master of Science in Physics and Mathematics thesis). Norwegian University of Science and Technology, Department of Mathematical Sciences.

31. Schneider, M., & Gama, N. (n.d.). SVP Challenge. Retrieved August 15, 2016, from `http://www.latticechallenge.org/svp-challenge/`.

32. Security Innovation (2014). NTRU Bibliography. Available at `https://assets.securityinnovation.com/static/downloads/NTRU/resources/NTRU-Bibliography.pdf`.

33. Security Innovation (n.d.). NTRU Cryptography & Signing. Available at https://assets.securityinnovation.com/static/downloads/datasheets/ntru-datasheet.pdf.

34. Security Innovation (n.d.). NTRU Post Quantum Cryptography. Retrieved August 15, 2016, from https://www.securityinnovation.com/products/ntru-crypto.

35. Security Innovation (n.d.). NTRU Resources. Retrieved August 15, 2016, from https://www.securityinnovation.com/products/ntru-crypto/ntru-resources.

36. Security Innovation (n.d.). NTRU Scrutiny. Retrieved August 15, 2016, from https://www.securityinnovation.com/products/ntru-crypto/ntru-scrutiny.

37. Security Innovation (n.d.). Open Source NTRU Public Key Cryptography Algorithm and Reference Code. Retrieved August 15, 2016, from https://github.com/NTRUOpenSourceProject/ntru-crypto/.

38. Security Innovation (n.d.). What is Post-Quantum Cryptography? Retrieved August 15, 2016, from https://web.securityinnovation.com/what-is-post-quantum-cryptography?

39. Shoup, V. (n.d.). NTL: A Library for doing Number Theory. Retrieved August 15, 2016, from http://www.shoup.net/ntl/.

40. Silverman, J. H. (1997). *Hard problems and backdoors for NTRU and others PKCS's* (Tech. Rep. No. 005). NTRU Cryptosystems.

41. Silverman, J. H. (1998). *Efficient conversions from Mod q to Mod p* (Tech. Rep. No. 008). NTRU Cryptosystems.

42. Silverman, J. H. (1998). *Invertibility in truncated polynomial rings* (Tech. Rep. No. 009). NTRU Cryptosystems.

43. Silverman, J. H. (1999). *Almost inverses and fast NTRU key creation* (Tech. Rep. No. 014). NTRU Cryptosystems.

44. Silverman, J. H. (1999). *Dimension-reduced lattices, zero-forced lattices and the NTRU public key cryptosystem* (Tech. Rep. No. 013). NTRU Cryptosystems.

45. Silverman, J. H. (1999). *High-speed multiplication of (truncated) polynomials* (Tech. Rep. No. 010). NTRU Cryptosystems.

46. Silverman, J. H. (2001). *Wraps, gaps, and lattice constants* (Tech. Rep. No. 011, Version 2). NTRU Cryptosystems.

47. Silverman, J. H., & Whyte, W. (2003). *Estimating decryption failure probabilities for NTRUEncrypt* (Tech. Rep. No. 018). NTRU Cryptosystems.

48. de Wolf, R. (2016). Lecture notes of the course *Quantum Computing*. University of Amsterdam.

49. Yasuda, T., Dahan, X., & Sakurai, K. (2015). Characterizing NTRU-variants using group ring and evaluating their lattice security. *IACR Cryptology ePrint Archive*, 2015/1170.